Cut your Cloud Costs by Half with Unikraft

Dr. Felipe Huici - CEO & Co-Founder Unikraft GmbH

The Private/Public Cloud

- Great scalability
- Easy to use
- Multitude of services

Cloud 1.0

The Private/Public Cloud

✓	✗
Great scalability	Bloated
Easy to use	Expensive
Multitude of services	Not eco-friendly
Cloud 1.0	**Current**

Good for massive cloud providers...

Good for massive cloud providers...

...and hardware manufacturers...

Good for massive cloud providers...

aws

Microsoft Azure

Google Cloud Platform

...and hardware manufacturers...

intel

SAMSUNG

AMD

...but not for most everyone else:

NETSUITE HarbourVest web.com SurveyMonkey opentext Hootsuite AUTODESK upland FrontierCapital

GE Capital mastercard box shopify Riverside Brookfield Warburg Pincus TPG BainCapital

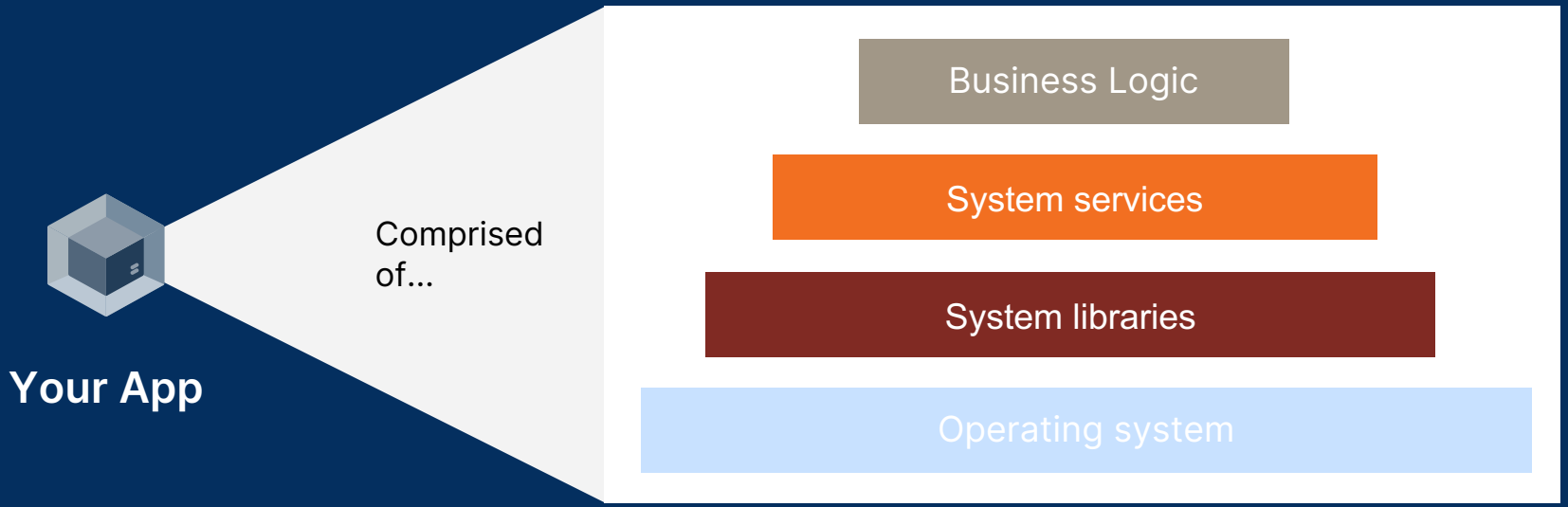
VISTA EQUITY PARTNERS Verifone The Carlyle Group workday TRAVELERS GANNETT JMI shutterstock

The Bloat Problem (pt.1)

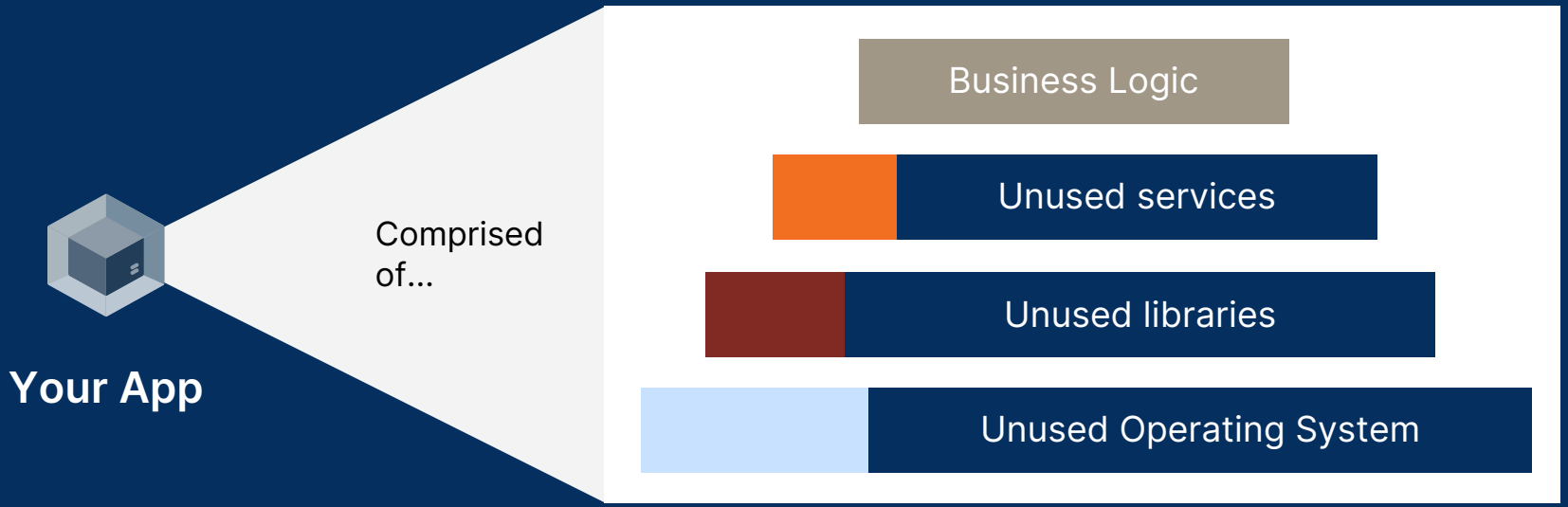
Your App

The Bloat Problem (pt.1)

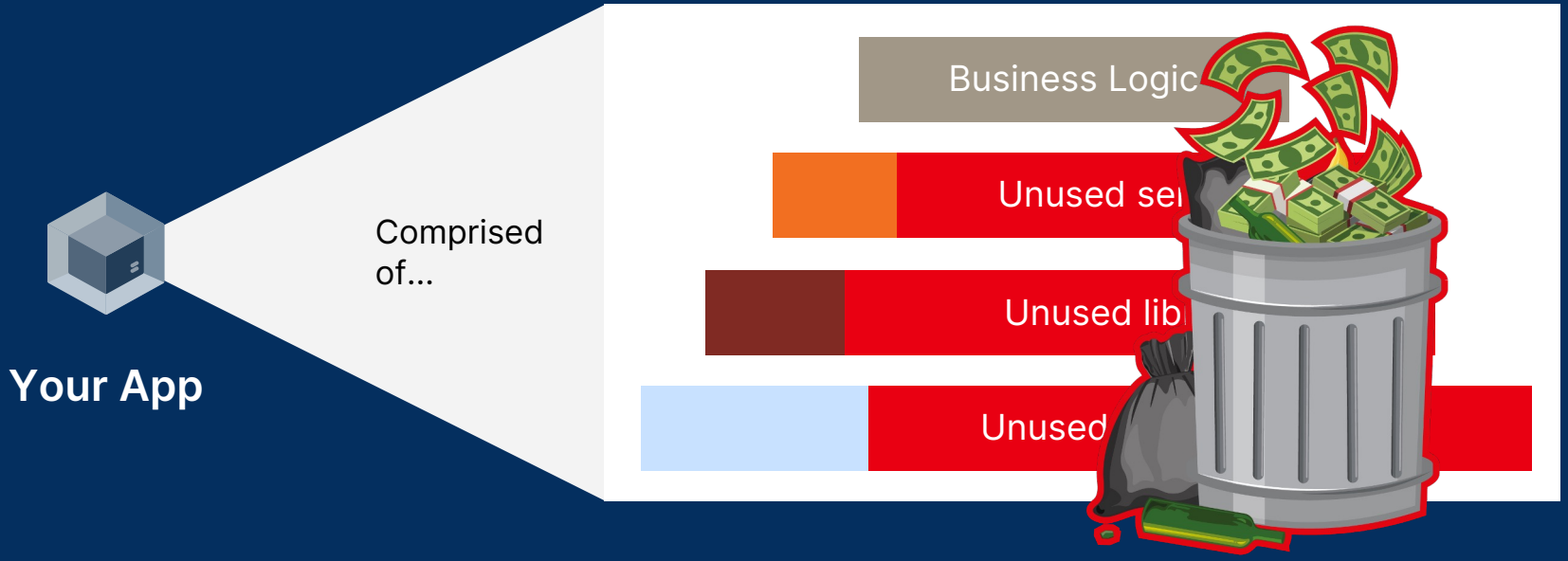
The Bloat Problem (pt.1)



The Bloat Problem (pt.1)



The Bloat Problem (pt.1)



The Bloat Problem (pt.2)

The Bloat Problem (pt.2)

The Bloat Problem (pt.2)

The Bloat Problem (pt.2)

The Bloat Problem (pt.2)

The Bloat Problem (pt.2)

Instance idling, wasting allocated resources

The Bloat Problem (pt.2)

Problem

You’re running a full café

Solution

You only ever want an espresso

Specialization

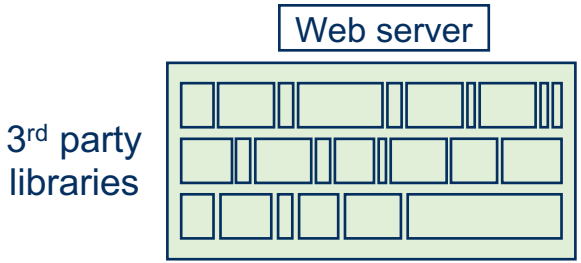
Specialization + Virtual Machines

Specialization + Virtual Machines

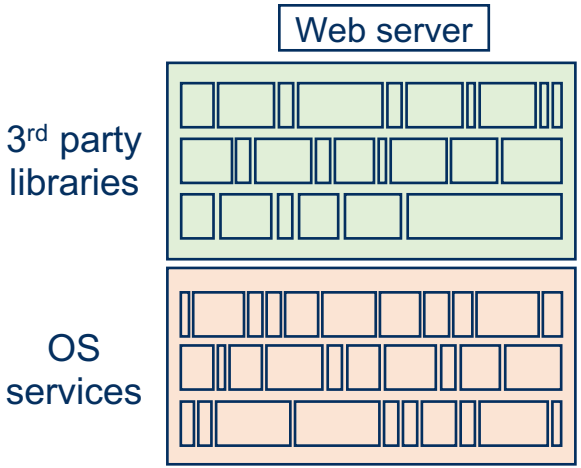
= Unikernels

Unikernels in a Nutshell

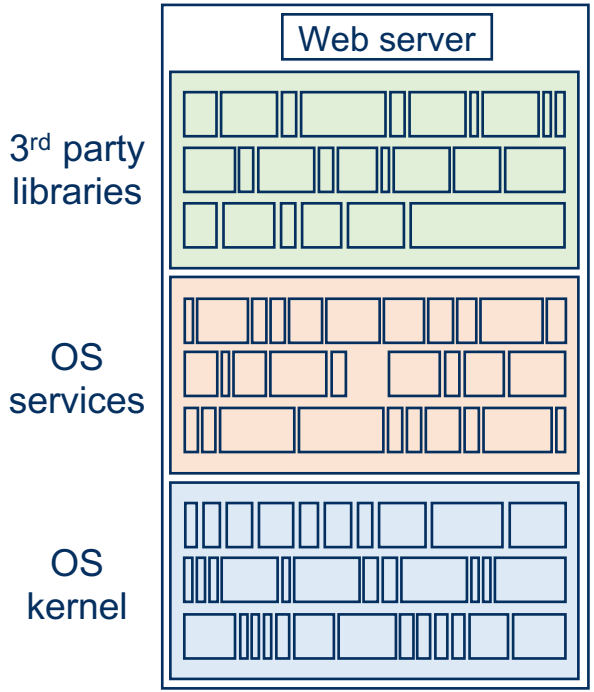
Unikernels in a Nutshell



Unikernels in a Nutshell

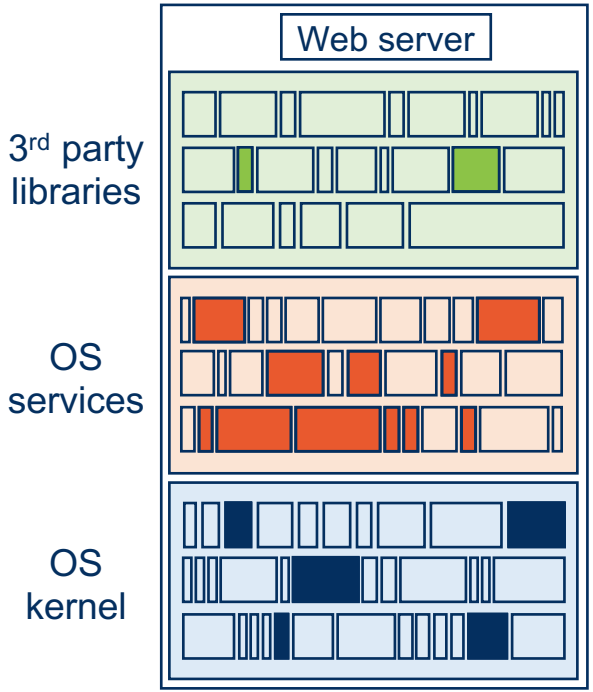


Unikernels in a Nutshell



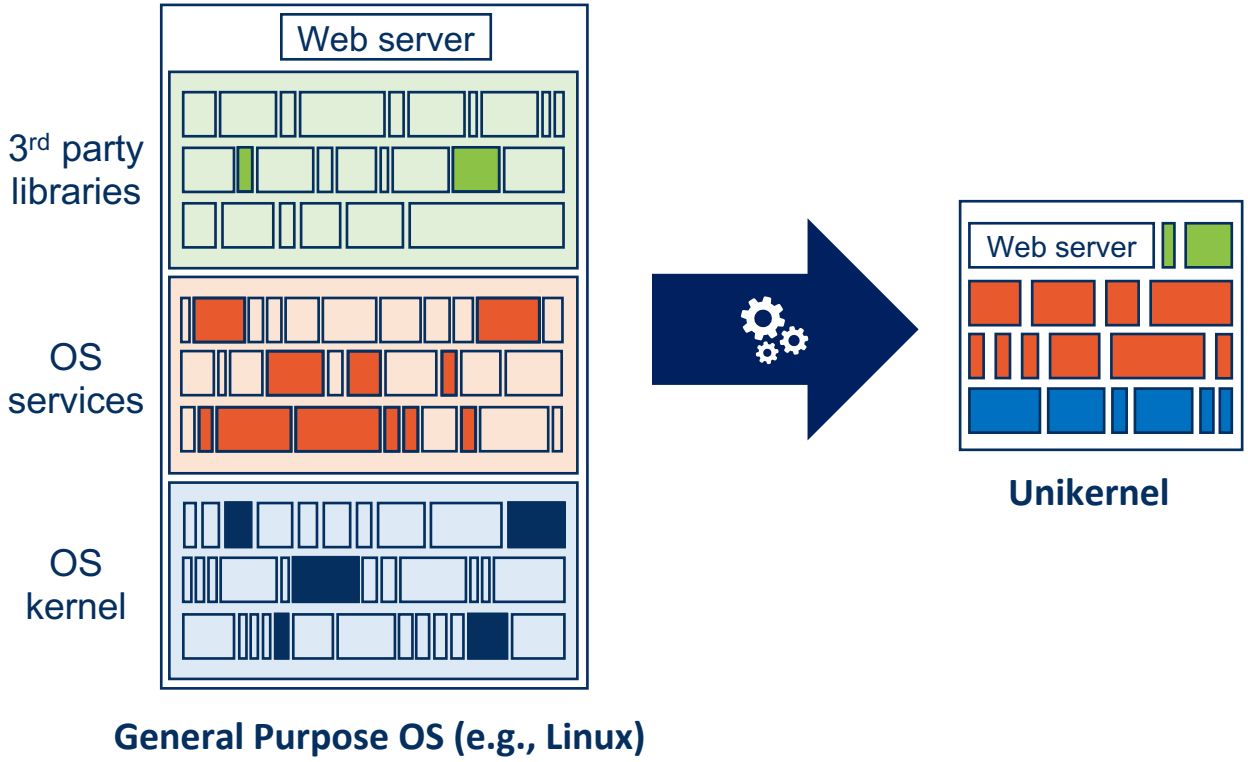
General Purpose OS (e.g., Linux)

Unikernels in a Nutshell

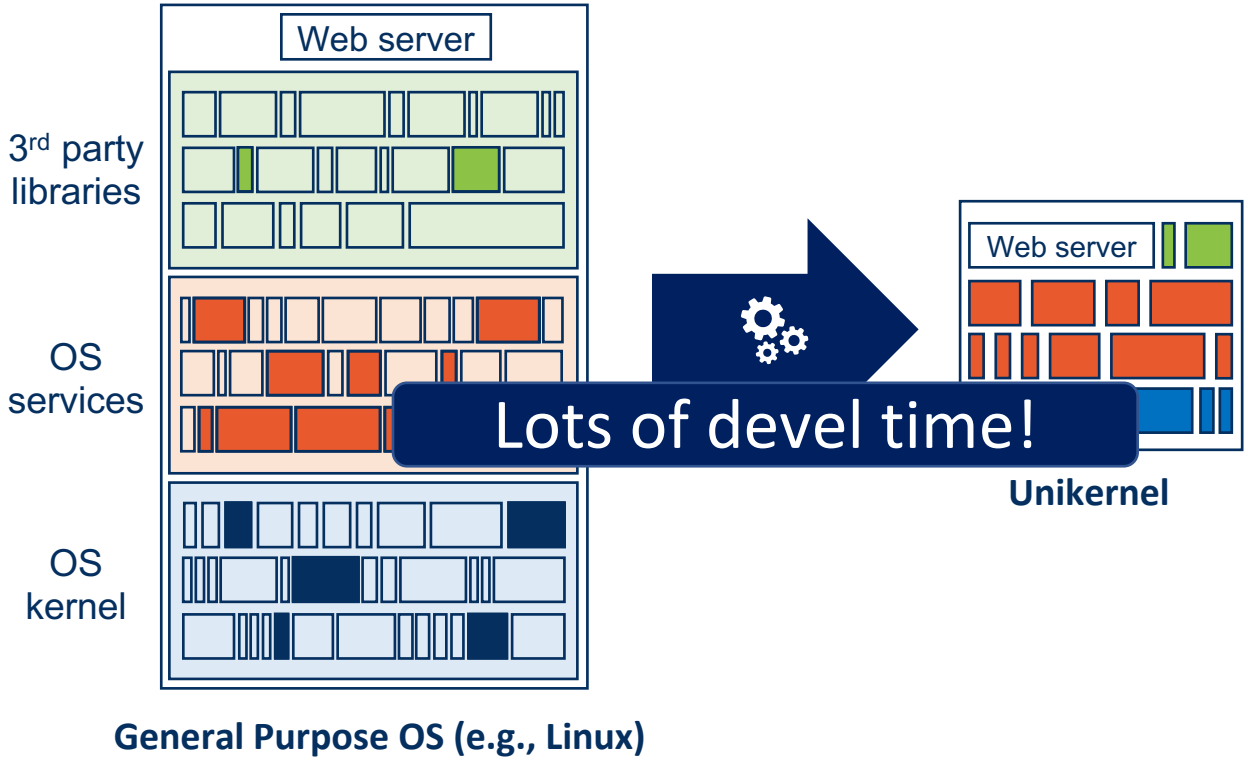


General Purpose OS (e.g., Linux)

Unikernels in a Nutshell



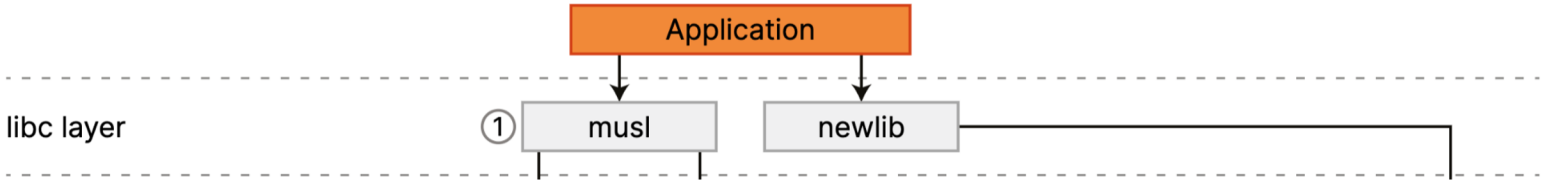
Unikernels in a Nutshell

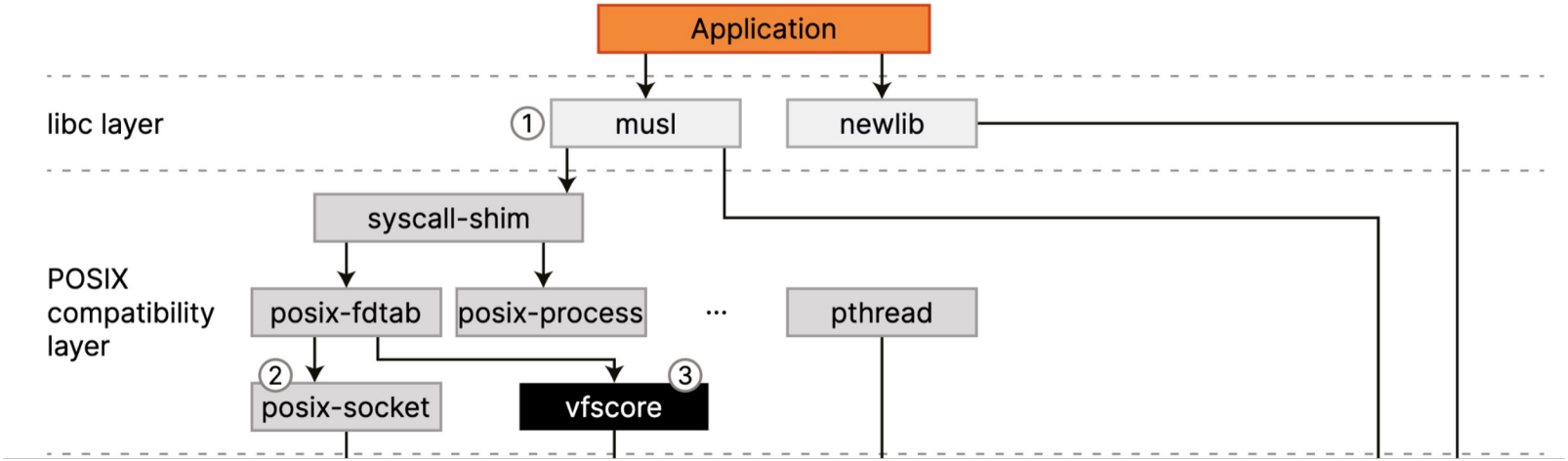


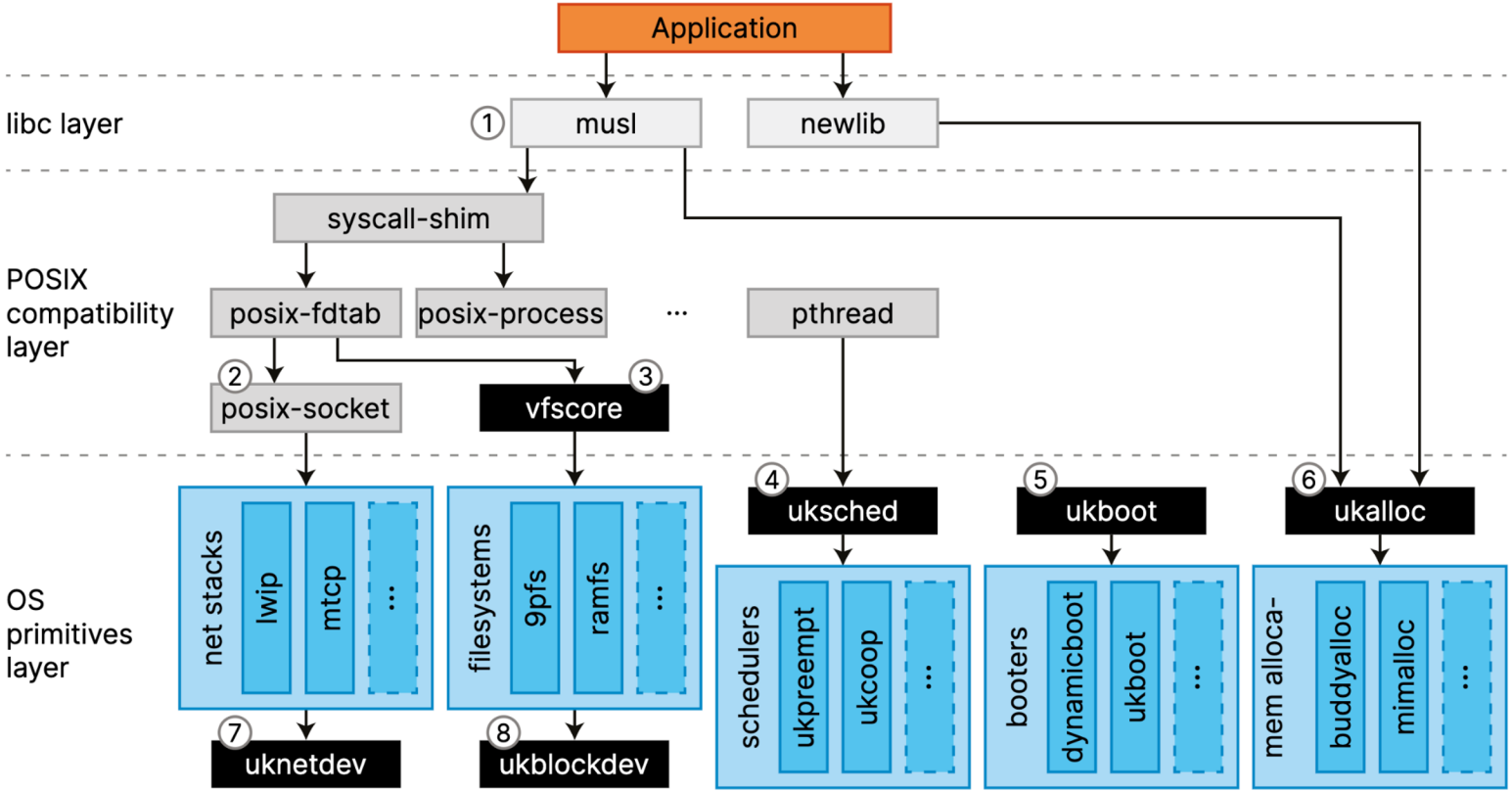
The Unikraft Model

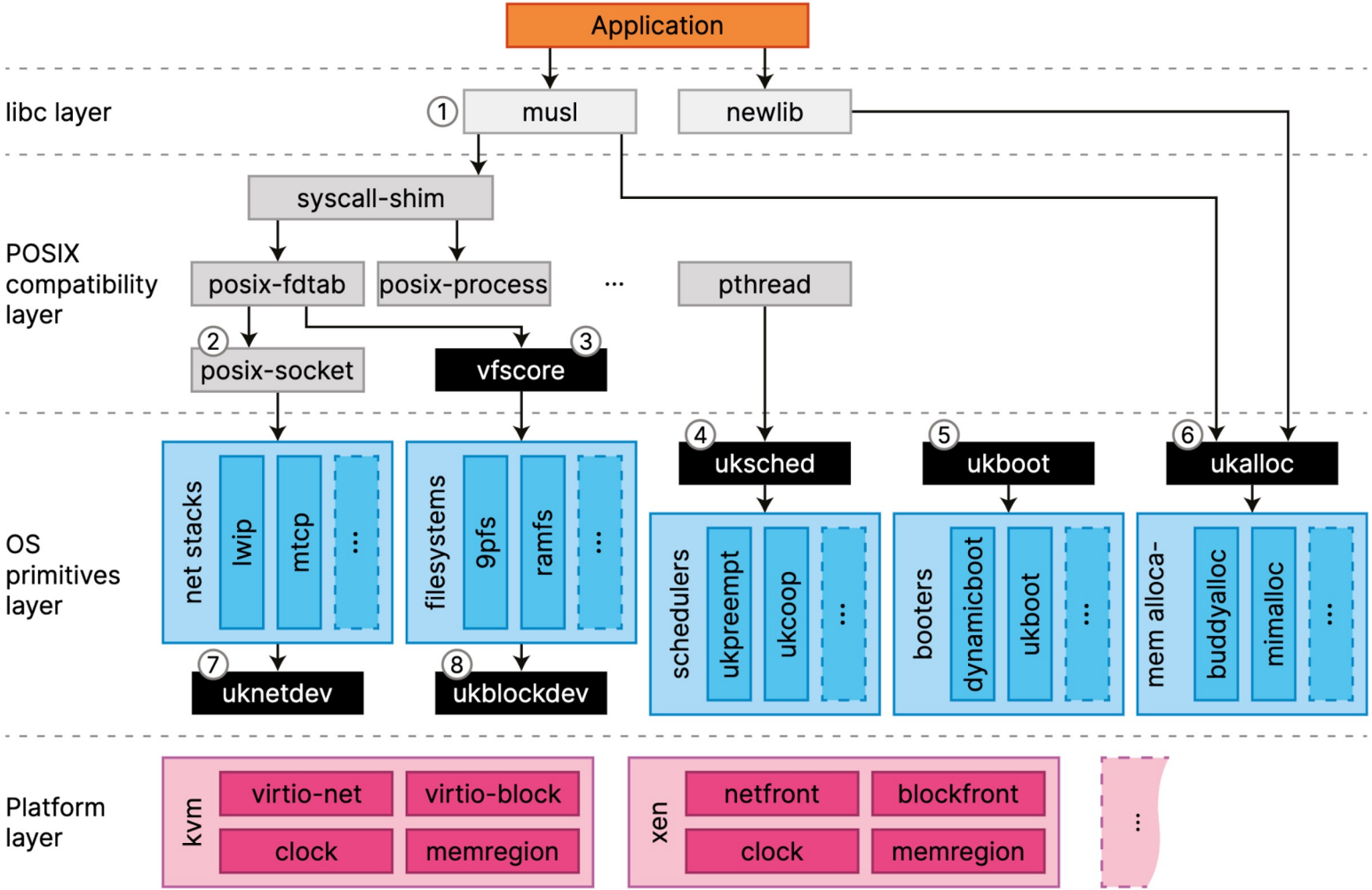
1. Fully modular architecture – everything as a library
2. Well-defined APIs
3. Core build system
4. Target POSIX compatibility

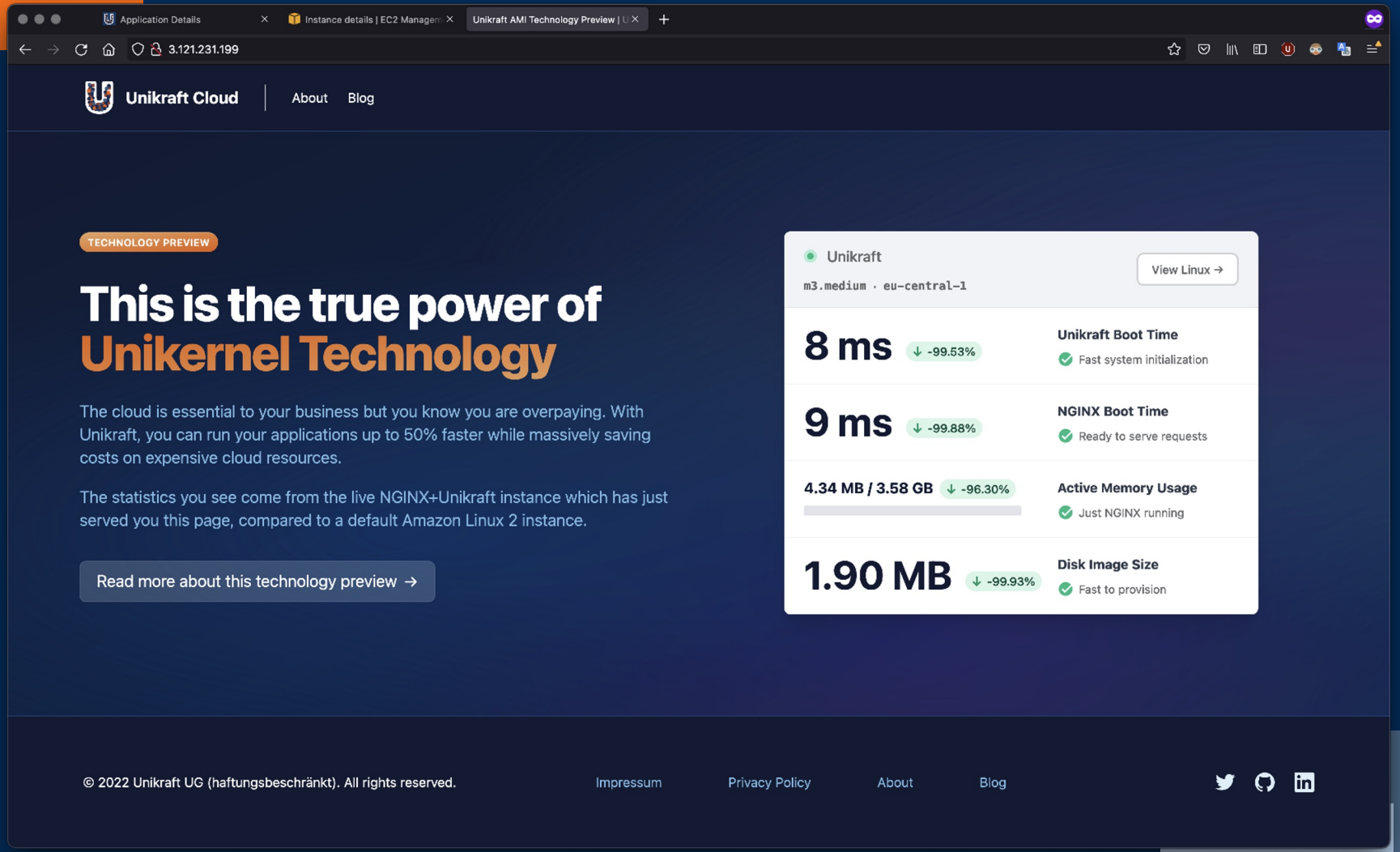

Application


Application
libc layer
1
musl
newlib


Application
libc layer
1
musl
newlib
syscall-shim
POSIX
compatibility
layer
posix-fdtab
posix-process
...
pthread
2
posix-socket
3
vfscore


Application
libc layer
1
musl
newlib
syscall-shim
POSIX
compatibility
layer
posix-fdtab
posix-process
...
pthread
2
posix-socket
3
vfscore
4
uksched
5
ukboot
6
ukalloc
OS
primitives
layer
net stacks
lwip
mtcp
...
filesystems
9pfs
ramfs
...
schedulers
ukpreempt
ukcoop
...
booters
dynamicboot
ukboot
...
mem alloca-
buddyalloc
mimalloc
...
7
uknetdev
8
ukblockdev


Application
libc layer
1
musl
newlib
syscall-shim
POSIX compatibility layer
posix-fdtab
posix-process
...
pthread
2
posix-socket
3
vfscore
4
uksched
5
ukboot
6
ukalloc
OS primitives layer
net stacks
lwip
mtcp
...
filesystems
9pfs
ramfs
...
schedulers
ukpreempt
ukcoop
...
booters
dynamicboot
ukboot
...
mem alloca-
buddyalloc
mimalloc
...
7
uknetdev
8
ukblockdev
Platform layer
kvm
virtio-net
virtio-block
clock
memregion
xen
netfront
blockfront
clock
memregion
...


Application Details
Instance details | EC2 Manage
Unikraft AMI Technology Preview |
3.121.231.199
Unikraft Cloud
About
Blog
TECHNOLOGY PREVIEW
This is the true power of
Unikernel Technology
The cloud is essential to your business but you know you are overpaying. With Unikraft, you can run your applications up to 50% faster while massively saving costs on expensive cloud resources.
The statistics you see come from the live NGINX+Unikraft instance which has just served you this page, compared to a default Amazon Linux 2 instance.
Read more about this technology preview →
Unikraft
m3.medium · eu-central-1
View Linux →
8 ms
-99.53%
Unikraft Boot Time
Fast system initialization
9 ms
-99.88%
NGINX Boot Time
Ready to serve requests
4.34 MB / 3.58 GB
-96.30%
Active Memory Usage
Just NGINX running
1.90 MB
-99.93%
Disk Image Size
Fast to provision
© 2022 Unikraft UG (haftungsbeschränkt). All rights reserved.
Impressum
Privacy Policy
About
Blog

Why aren’t unikernels widely deployed?

Barriers to Deployment

1. Performance
2. Application & platform support
3. Framework integration
4. Debugging

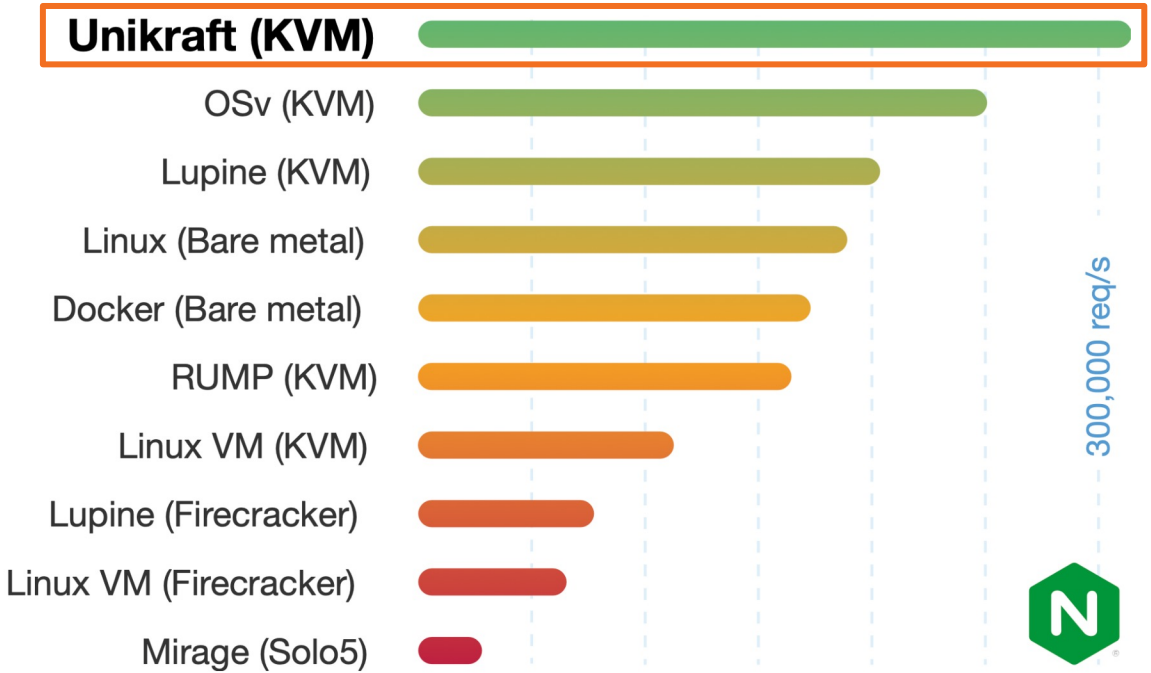
Barriers to Deployment

1. Performance
2. Application & platform support
3. Framework integration
4. Debugging

Barriers to Deployment

1. Performance
    a. Small but still monolithic
    b. Underperforming code
    c. “Slow” languages

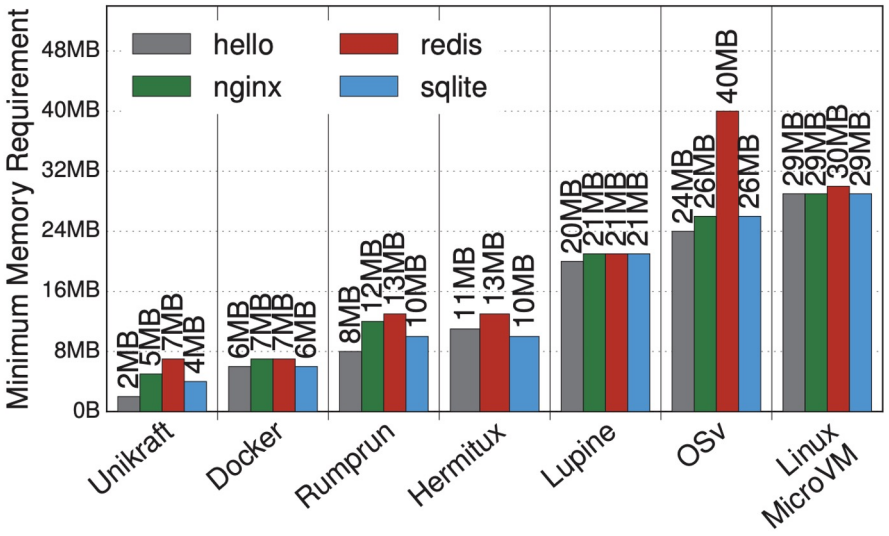
Unikraft provides better performance



On Unikraft, **82%** of increase compared to Docker (NGINX throughput)

Unikraft provides better memory consumption and storage

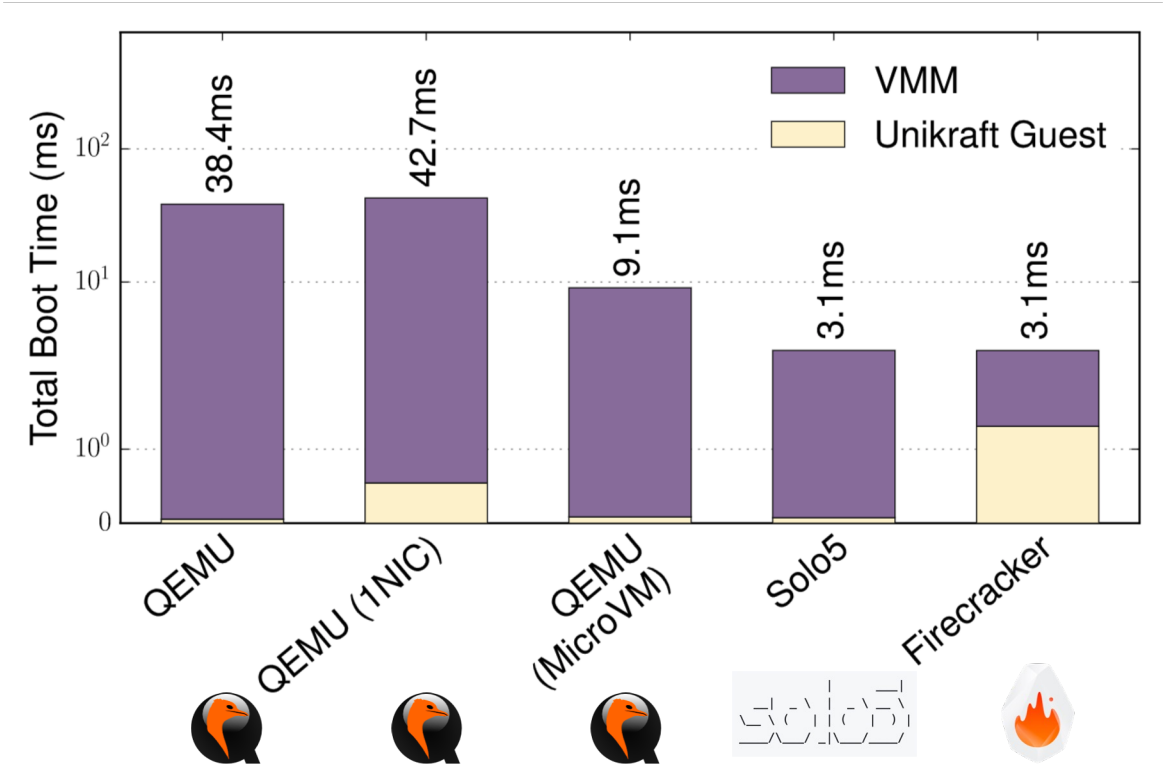
Memory Usage



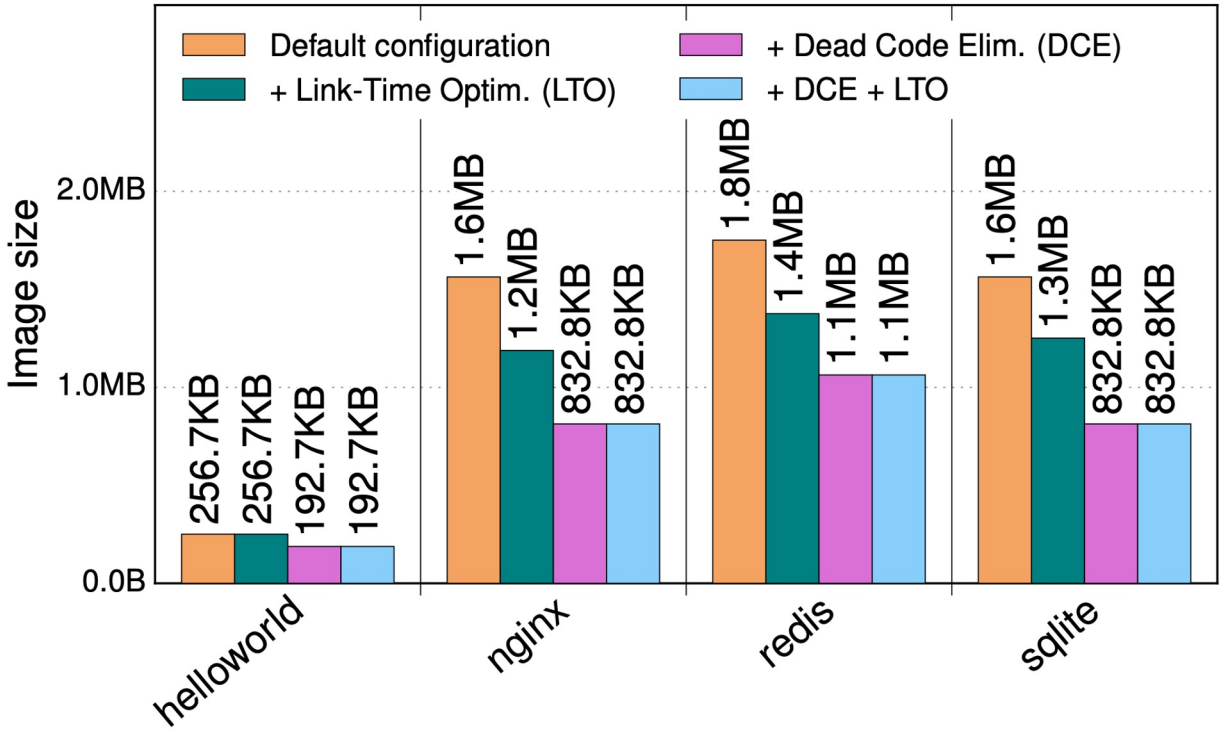
Disk Space

Image	Size
`docker.io/nginx:1.15.6`	42.62 MB
`unikraft.io/nginx:1.15.6`	1.3 MB

Performance on varying Virtual Machine Monitors



Unikraft offers better optimization

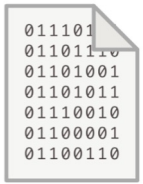


Barriers to Deployment

1. ~~Performance~~
2. Application & platform support
    a. Language-specific unikernels
    b. Application porting required
    c. Insufficient syscall support
3. Framework integration
4. Debugging

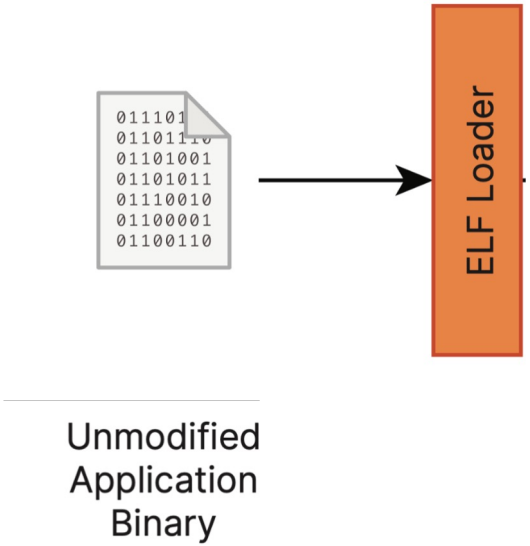
What Unikraft Supports (Sample)

POSIX Compatibility - Binary Mode

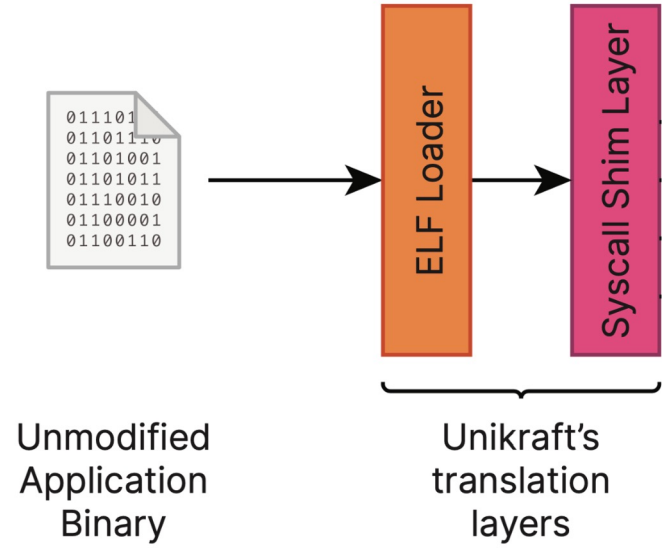


Unmodified Application Binary

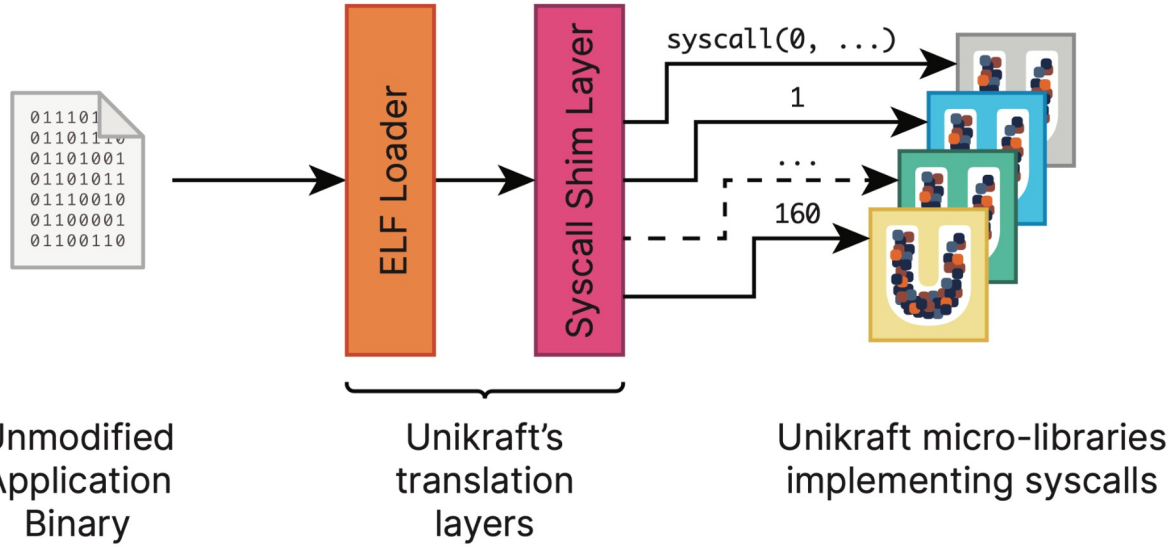
POSIX Compatibility - Binary Mode



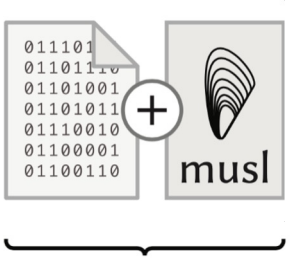
POSIX Compatibility - Binary Mode



POSIX Compatibility - Binary Mode

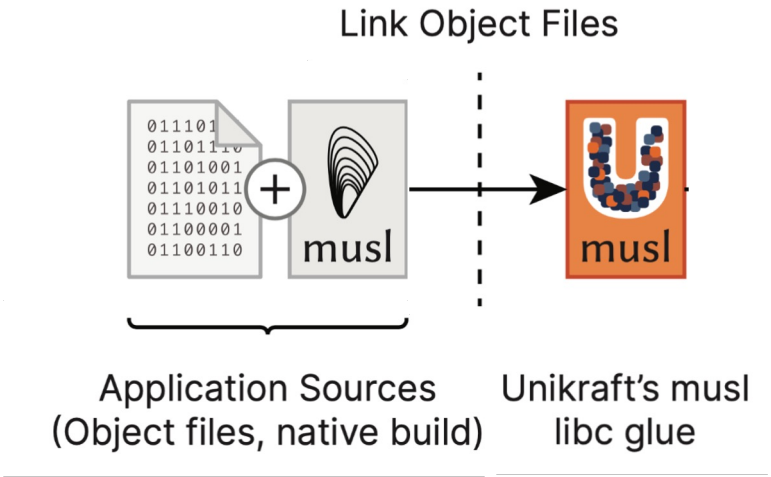


POSIX Compatibility - Source/Musl (WiP)

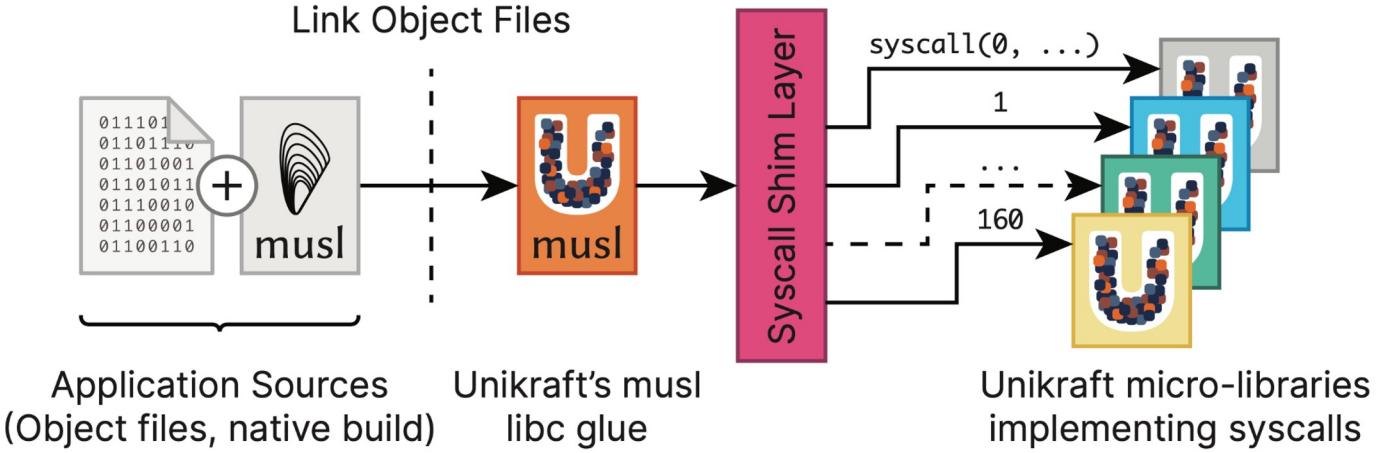


Application Sources
(Object files, native build)

POSIX Compatibility - Source/Musl (WiP)



POSIX Compatibility - Source/Musl (WiP)



Platform (and libc) Support

libc: nolibc

hyper: Xen Project

arch: intel

Platform (and libc) Support

libc:

hyper:

arch:

Platform (and libc) Support

libc: nolibc newlib

hyper: Xen Project KVM

arch: arm intel

Platform (and libc) Support

libc:

nolibc

hyper:

arch:

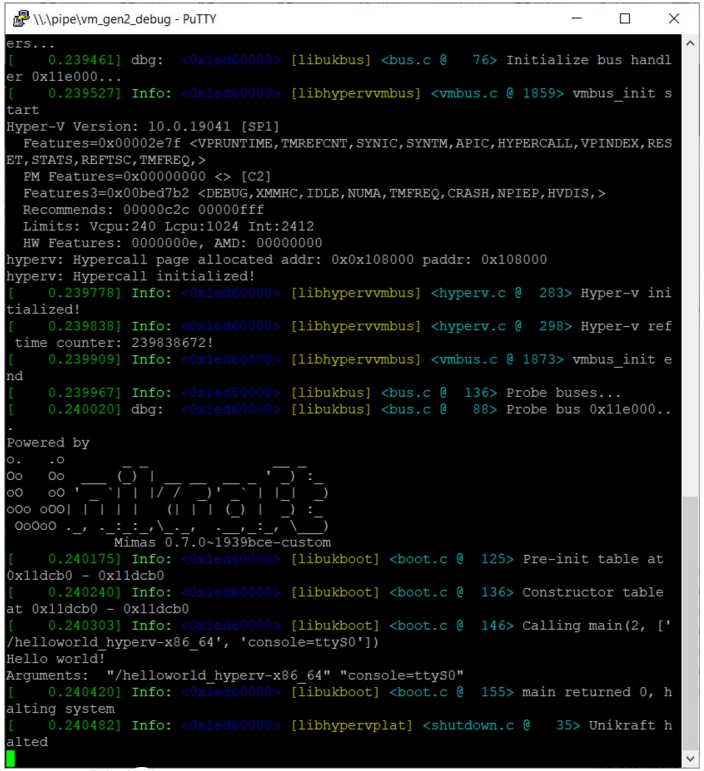
Platform (and libc) Support

libc:

nolibc

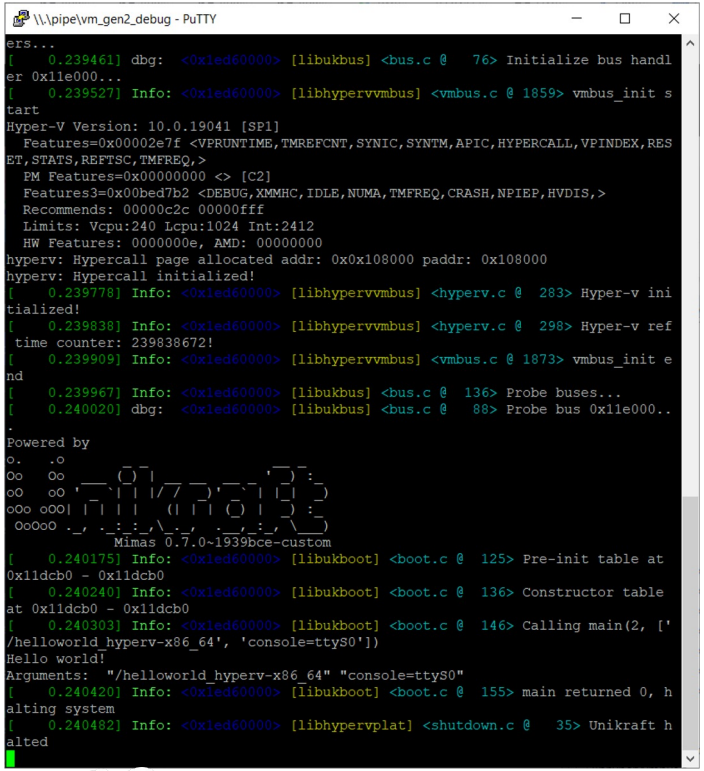
hyper:

arch:



Unikraft +

Microsoft Hyper-V

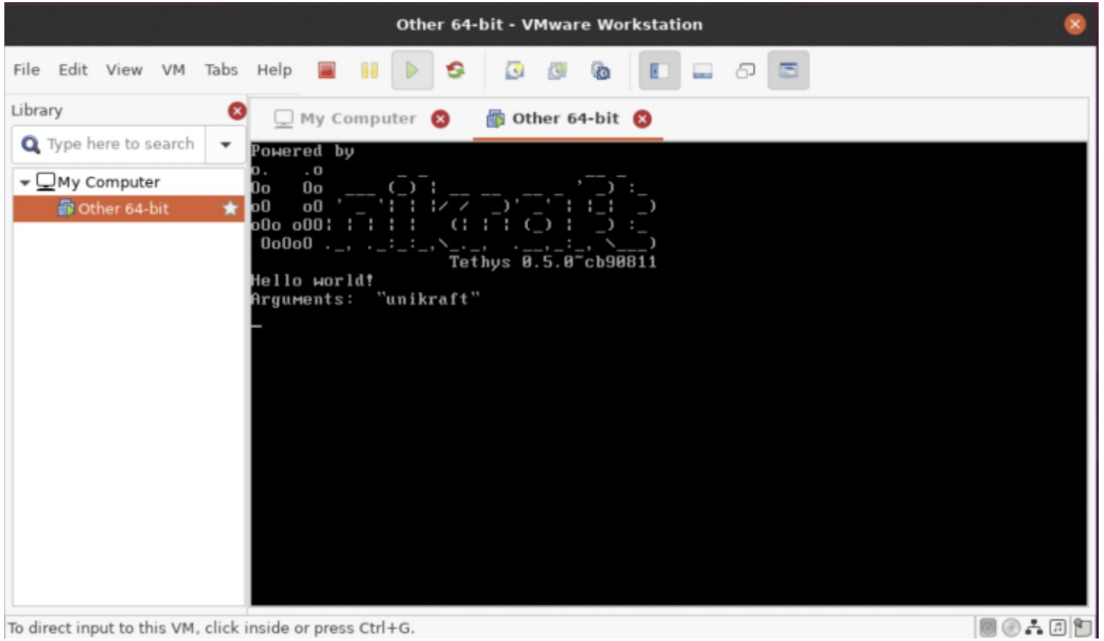


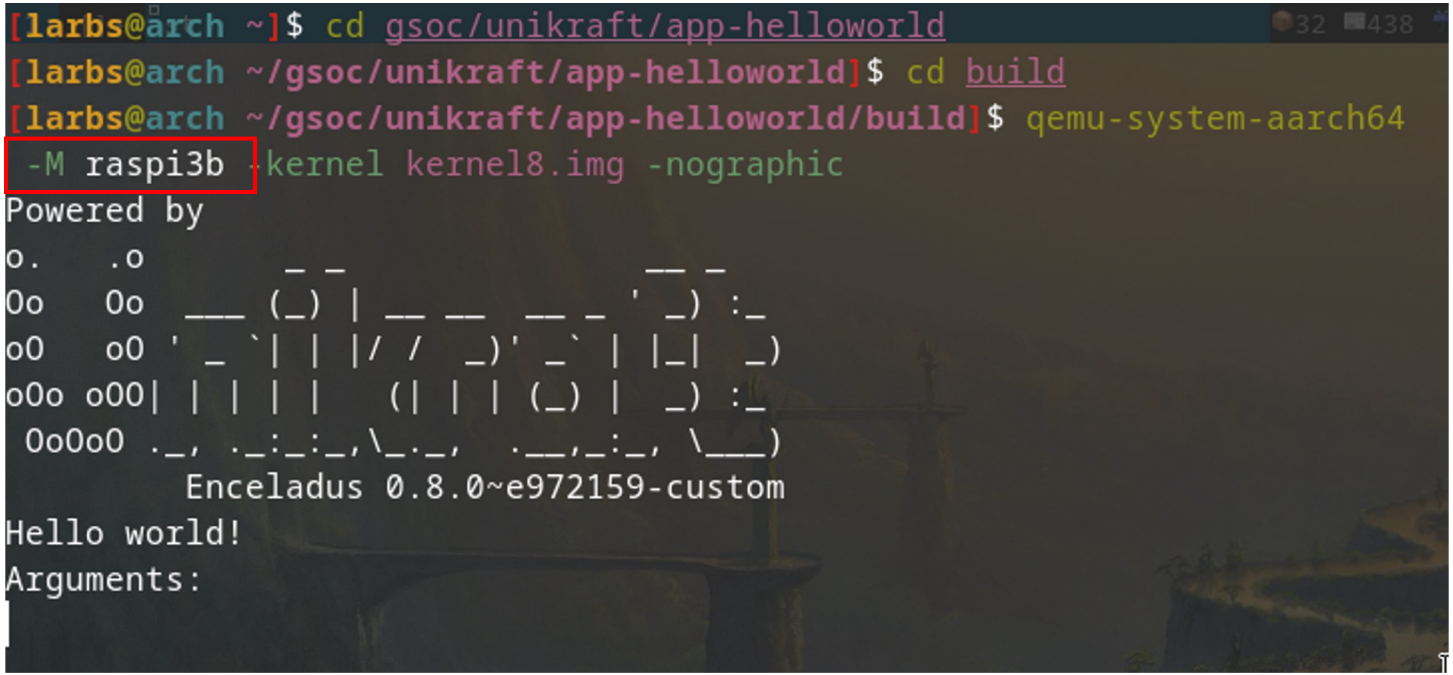
Unikraft +

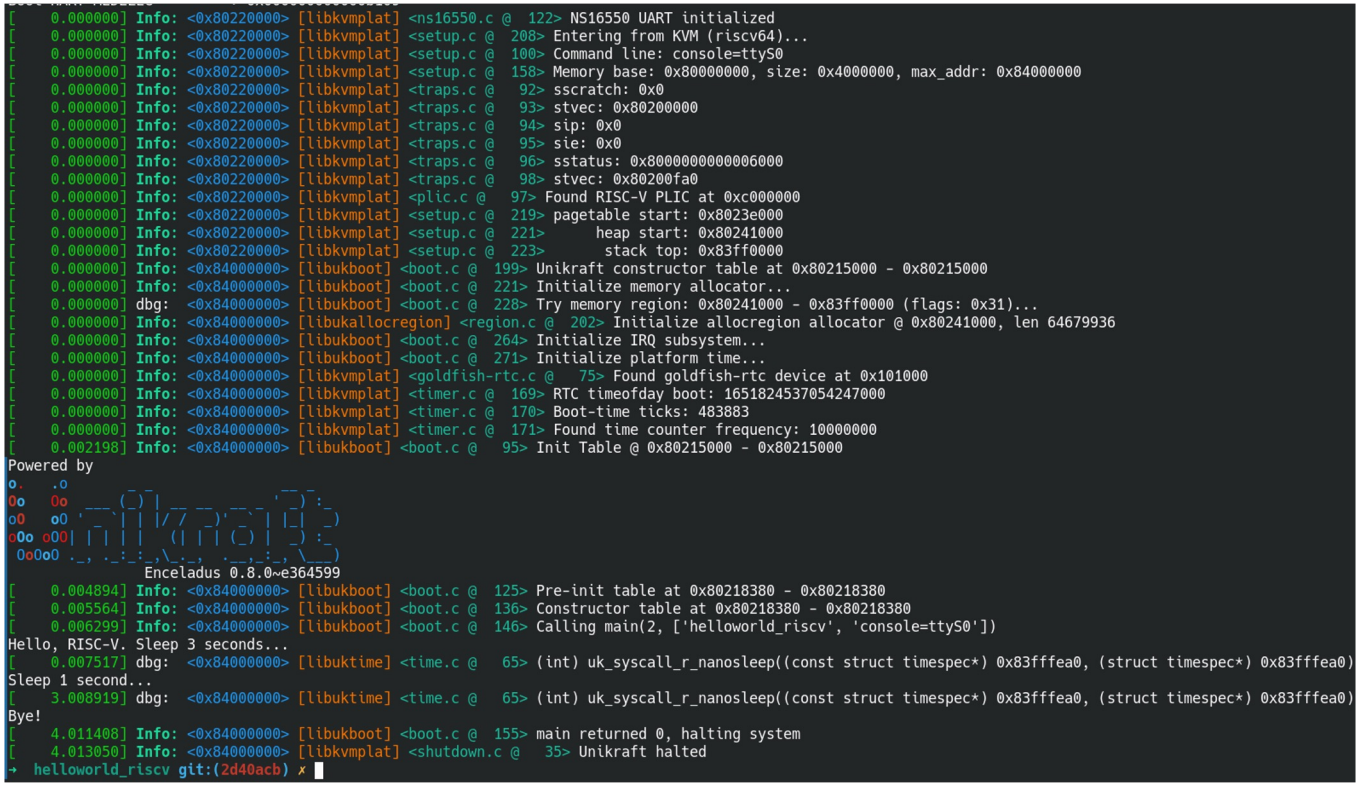
Microsoft Hyper-V

Unikraft +

vmware®




[larbs@arch ~]$ cd gsoc/unikraft/app-helloworld
[larbs@arch ~/gsoc/unikraft/app-helloworld]$ cd build
[larbs@arch ~/gsoc/unikraft/app-helloworld/build]$ qemu-system-aarch64
-M raspi3b -kernel kernel8.img -nographic
Powered by
Enceladus 0.8.0~e972159-custom
Hello world!
Arguments:


[0.000000] Info: <0x80220000> [libkvmplat] <ns16550.c @ 122> NS16550 UART initialized
[0.000000] Info: <0x80220000> [libkvmplat] <setup.c @ 208> Entering from KVM (riscv64)...
[0.000000] Info: <0x80220000> [libkvmplat] <setup.c @ 100> Command line: console=ttyS0
[0.000000] Info: <0x80220000> [libkvmplat] <setup.c @ 158> Memory base: 0x80000000, size: 0x4000000, max_addr: 0x84000000
[0.000000] Info: <0x80220000> [libkvmplat] <traps.c @ 92> sscratch: 0x0
[0.000000] Info: <0x80220000> [libkvmplat] <traps.c @ 93> stvec: 0x80200000
[0.000000] Info: <0x80220000> [libkvmplat] <traps.c @ 94> sip: 0x0
[0.000000] Info: <0x80220000> [libkvmplat] <traps.c @ 95> sie: 0x0
[0.000000] Info: <0x80220000> [libkvmplat] <traps.c @ 96> sstatus: 0x8000000000006000
[0.000000] Info: <0x80220000> [libkvmplat] <traps.c @ 98> stvec: 0x80200fa0
[0.000000] Info: <0x80220000> [libkvmplat] <plic.c @ 97> Found RISC-V PLIC at 0xc000000
[0.000000] Info: <0x80220000> [libkvmplat] <setup.c @ 219> pagetable start: 0x8023e000
[0.000000] Info: <0x80220000> [libkvmplat] <setup.c @ 221> heap start: 0x80241000
[0.000000] Info: <0x80220000> [libkvmplat] <setup.c @ 223> stack top: 0x83ff0000
[0.000000] Info: <0x84000000> [libukboot] <boot.c @ 199> Unikraft constructor table at 0x80215000 - 0x80215000
[0.000000] Info: <0x84000000> [libukboot] <boot.c @ 221> Initialize memory allocator...
[0.000000] dbg: <0x84000000> [libukboot] <boot.c @ 228> Try memory region: 0x80241000 - 0x83ff0000 (flags: 0x31)...
[0.000000] Info: <0x84000000> [libukallocregion] <region.c @ 202> Initialize allocregion allocator @ 0x80241000, len 64679936
[0.000000] Info: <0x84000000> [libukboot] <boot.c @ 264> Initialize IRQ subsystem...
[0.000000] Info: <0x84000000> [libukboot] <boot.c @ 271> Initialize platform time...
[0.000000] Info: <0x84000000> [libkvmplat] <goldfish-rtc.c @ 75> Found goldfish-rtc device at 0x101000
[0.000000] Info: <0x84000000> [libkvmplat] <timer.c @ 169> RTC timeofday boot: 1651824537054247000
[0.000000] Info: <0x84000000> [libkvmplat] <timer.c @ 170> Boot-time ticks: 483883
[0.000000] Info: <0x84000000> [libkvmplat] <timer.c @ 171> Found time counter frequency: 10000000
[0.002198] Info: <0x84000000> [libukboot] <boot.c @ 95> Init Table @ 0x80215000 - 0x80215000
Powered by
Enceladus 0.8.0~e364599
[0.004894] Info: <0x84000000> [libukboot] <boot.c @ 125> Pre-init table at 0x80218380 - 0x80218380
[0.005564] Info: <0x84000000> [libukboot] <boot.c @ 136> Constructor table at 0x80218380 - 0x80218380
[0.006299] Info: <0x84000000> [libukboot] <boot.c @ 146> Calling main(2, ['helloworld_riscv', 'console=ttyS0'])
Hello, RISC-V. Sleep 3 seconds...
[0.007517] dbg: <0x84000000> [libuktime] <time.c @ 65> (int) uk_syscall_r_nanosleep((const struct timespec*) 0x83fffea0, (struct timespec*) 0x83fffea0)
Sleep 1 second...
[3.008919] dbg: <0x84000000> [libuktime] <time.c @ 65> (int) uk_syscall_r_nanosleep((const struct timespec*) 0x83fffea0, (struct timespec*) 0x83fffea0)
Bye!
[4.011408] Info: <0x84000000> [libukboot] <boot.c @ 155> main returned 0, halting system
[4.013050] Info: <0x84000000> [libkvmplat] <shutdown.c @ 35> Unikraft halted
helloworld_riscv git:(2d40acb) x

Barriers to Deployment

1. ~~Performance~~
2. ~~Application & platform support~~
3. Framework integration
    a. Inexistent standard frameworks
    b. Missing toolkits
4. Debugging

Seamless Deployment & Integration

VSCode

Easy development on the most popular IDE platform

kraft

Easily build your unikraft unikernel

Kubernetes

Deploy extremely efficient Unikraft images seamless against your Kubernetes cluster

Prometheus

Monitor your Unikraft instances through a standard and state-of-the-art monitoring platform

Seamless Deployment & Integration

VSCode

Easy development on the most popular IDE platform

kraft

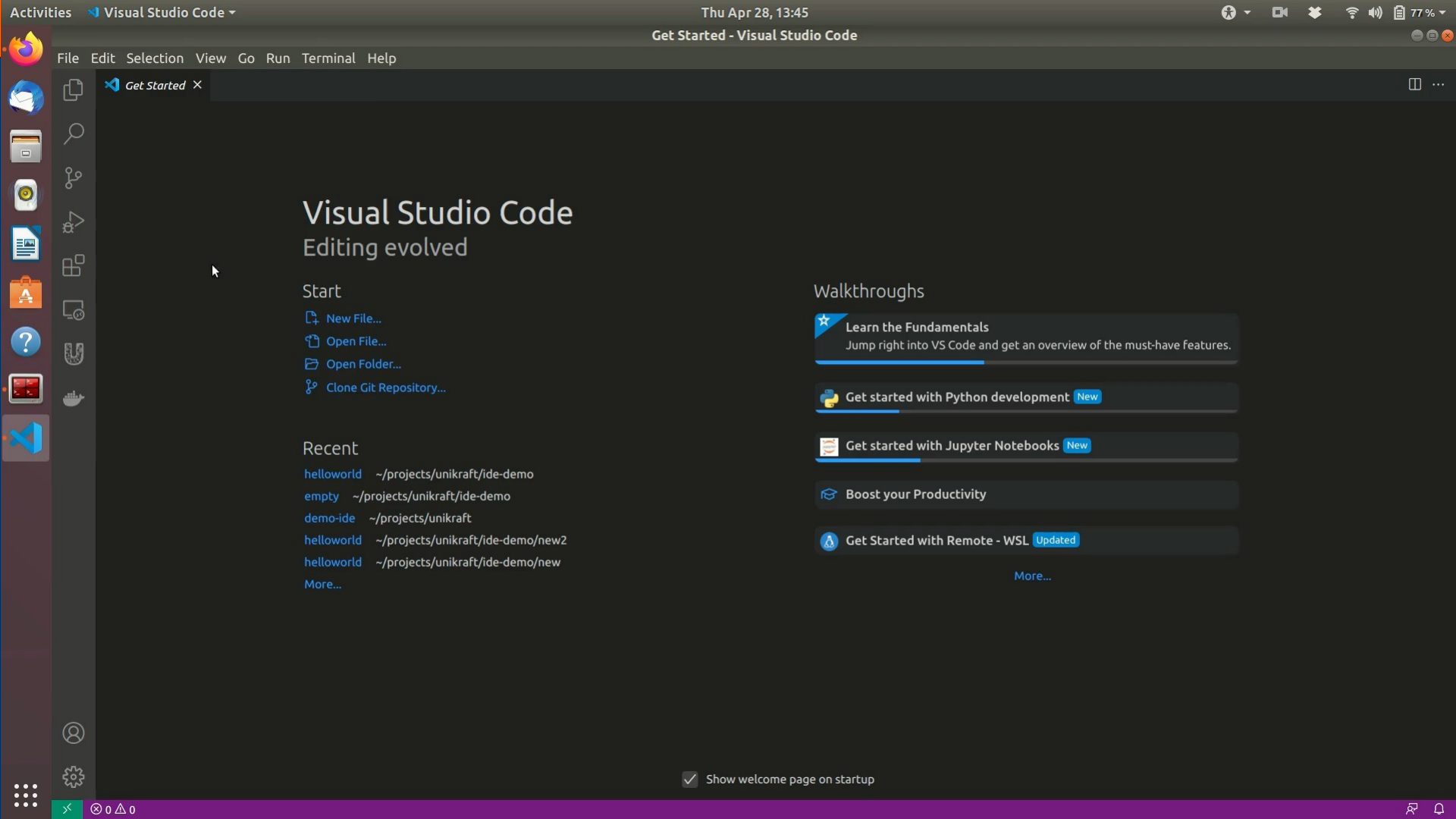
Easily build your unikraft unikernel

Kubernetes

Deploy extremely efficient Unikraft images seamless against your Kubernetes cluster

Prometheus

Monitor your Unikraft instances through a standard and state-of-the-art monitoring platform


Activities
Visual Studio Code
Thu Apr 28, 13:45
77 %
Get Started - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Get Started
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Recent
helloworld ~/projects/unikraft/ide-demo
empty ~/projects/unikraft/ide-demo
demo-ide ~/projects/unikraft
helloworld ~/projects/unikraft/ide-demo/new2
helloworld ~/projects/unikraft/ide-demo/new
More...
Walkthroughs
Learn the Fundamentals
Jump right into VS Code and get an overview of the must-have features.
Get started with Python development New
Get started with Jupyter Notebooks New
Boost your Productivity
Get Started with Remote - WSL Updated
More...
Show welcome page on startup
0 0

Seamless Deployment & Integration

VSCode

Easy development on the most popular IDE platform

kraft

Easily build your unikraft unikernel

Kubernetes

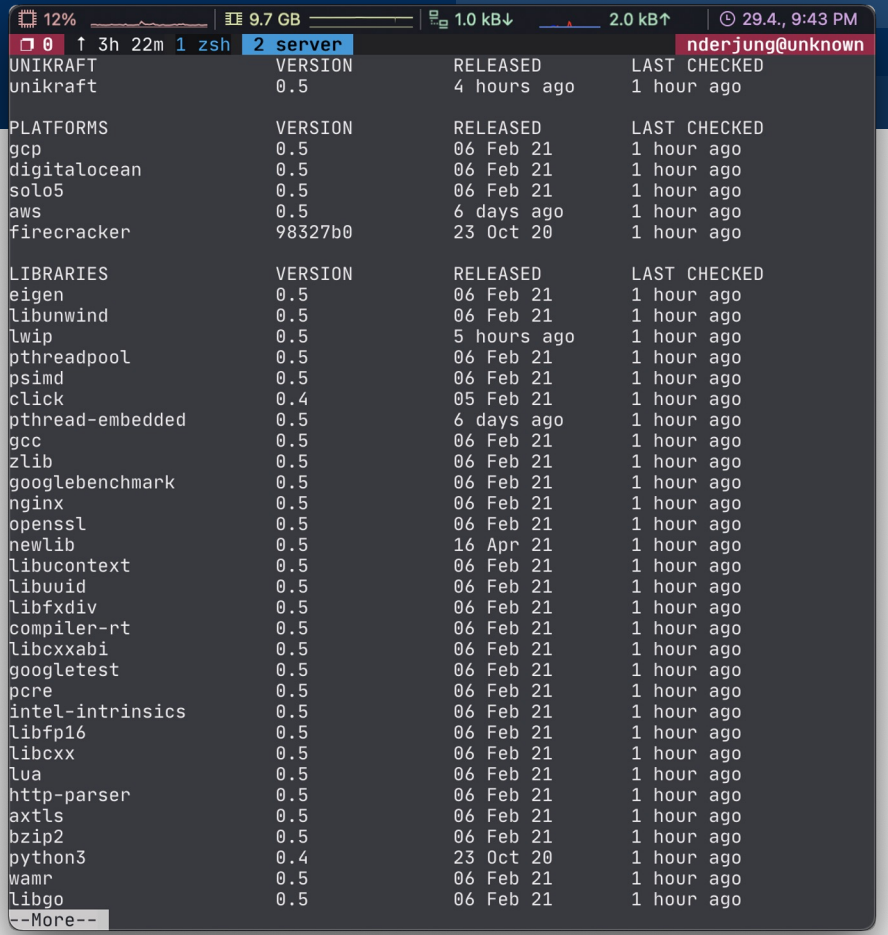
Deploy extremely efficient Unikraft images seamless against your Kubernetes cluster

Prometheus

Monitor your Unikraft instances through a standard and state-of-the-art monitoring platform

kraft

- Easily manage multiple libraries from different sources
- Quickly access updates and change between versions
- Automatically download application source dependencies




root@kraft:~#
Unikraft
[0] 0:bash*
kraft NGINX demo

Seamless Deployment & Integration

VSCode

Easy development on the most popular IDE platform

kraft

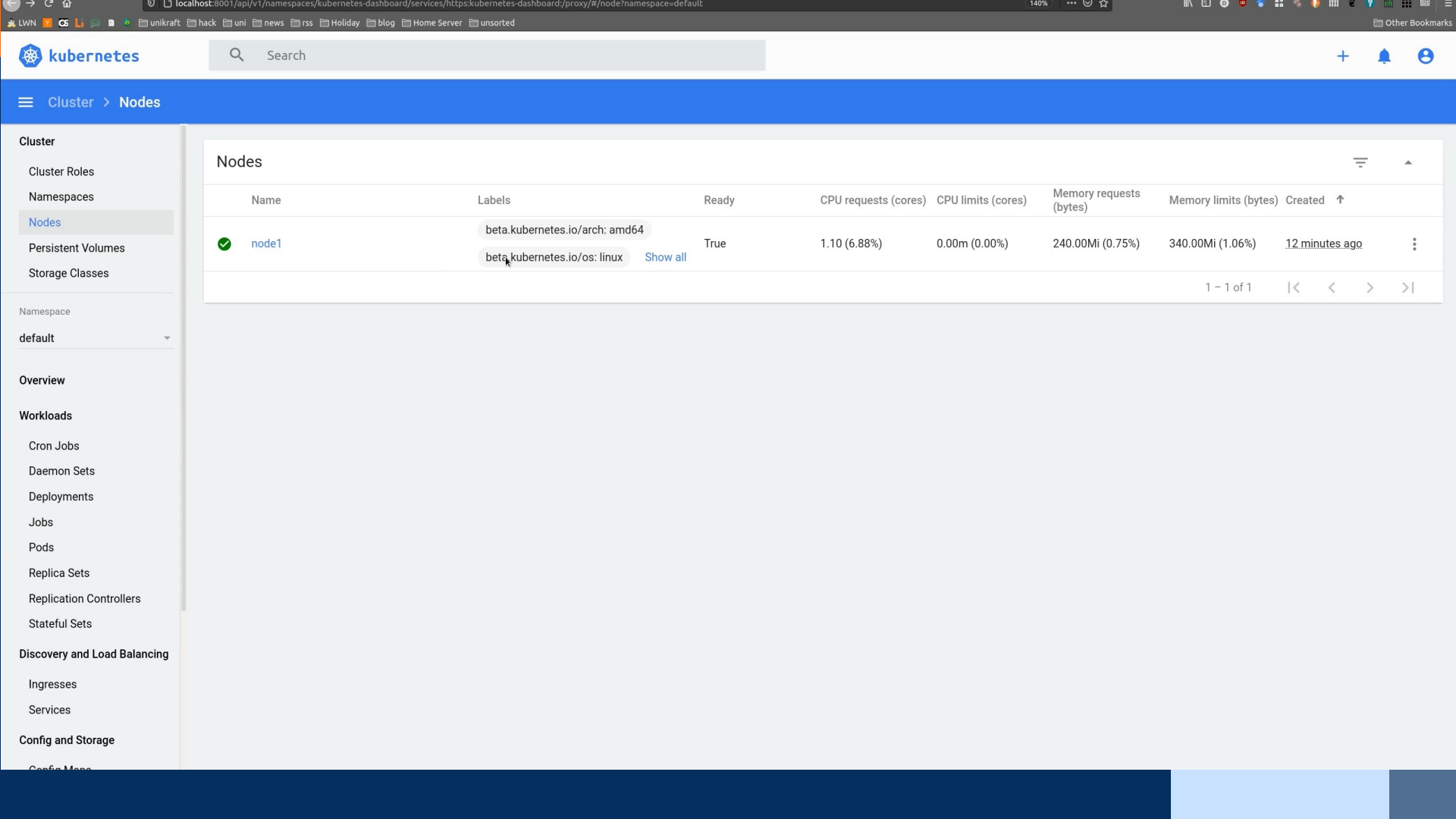
Easily build your unikraft unikernel

Kubernetes

Deploy extremely efficient Unikraft images seamless against your Kubernetes cluster

Prometheus

Monitor your Unikraft instances through a standard and state-of-the-art monitoring platform


localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/#/node?namespace=default
kubernetes
Search
Cluster > Nodes
Cluster
Cluster Roles
Namespaces
Nodes
Persistent Volumes
Storage Classes
Namespace
default
Overview
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Discovery and Load Balancing
Ingresses
Services
Config and Storage
Nodes
Name
Labels
Ready
CPU requests (cores)
CPU limits (cores)
Memory requests (bytes)
Memory limits (bytes)
Created
node1
beta.kubernetes.io/arch: amd64
beta.kubernetes.io/os: linux
Show all
True
1.10 (6.88%)
0.00m (0.00%)
240.00Mi (0.75%)
340.00Mi (1.06%)
12 minutes ago
1 – 1 of 1

Seamless Deployment & Integration

VSCode

Easy development on the most popular IDE platform

kraft

Easily build your unikraft unikernel

Kubernetes

Deploy extremely efficient Unikraft images seamless against your Kubernetes cluster

Prometheus

Monitor your Unikraft instances through a standard and state-of-the-art monitoring platform

Unikraft Prometheus Exporter + Grafana Dashboard

Barriers to Deployment

~~1. Performance~~

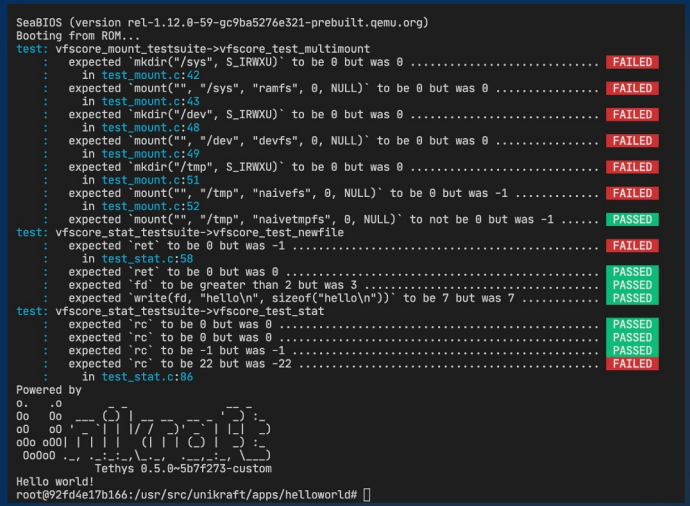
~~2. Application & platform support~~

~~3. Framework integration~~

4. Debugging
    a. No standard tools (e.g., gdb)
    b. No profiling tools
    c. Poor or no documentation

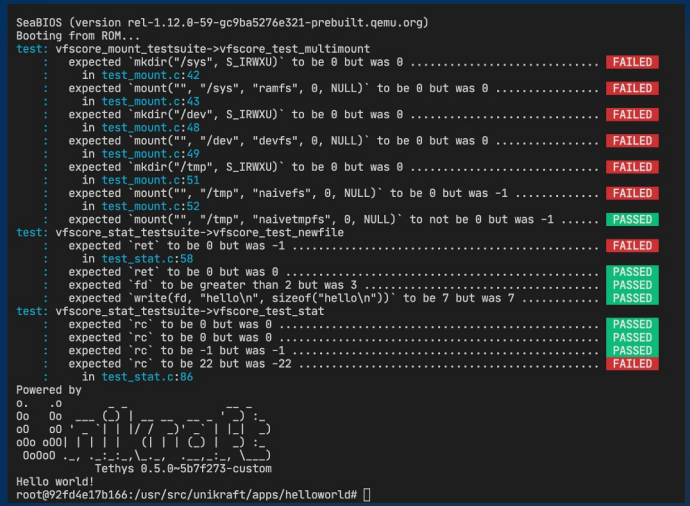
Monitoring & Debugging Features

- ukdebug
  - Logging/Print system
  - Assertions
  - Tracepoints
  - GDB server



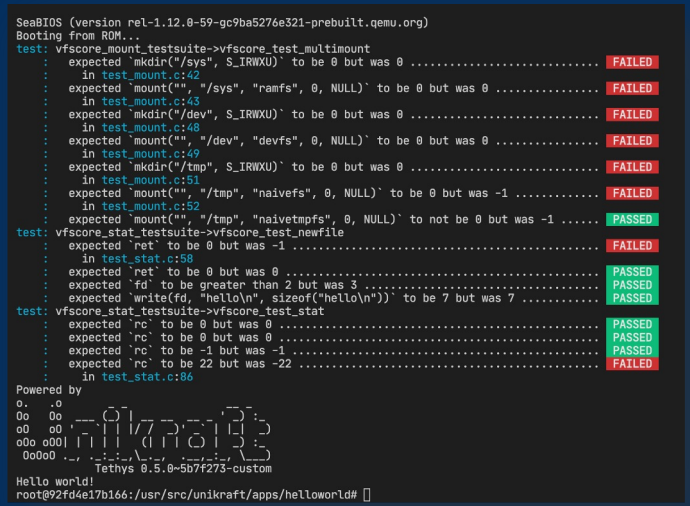
Monitoring & Debugging Features

- ukdebug
  - Logging/Print system
  - Assertions
  - Tracepoints
  - GDB server
- `uktest`
  - Unit Testing



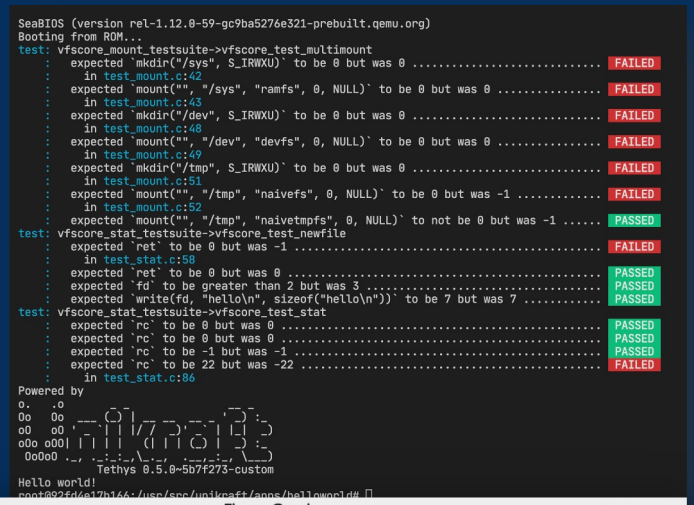
Monitoring & Debugging Features

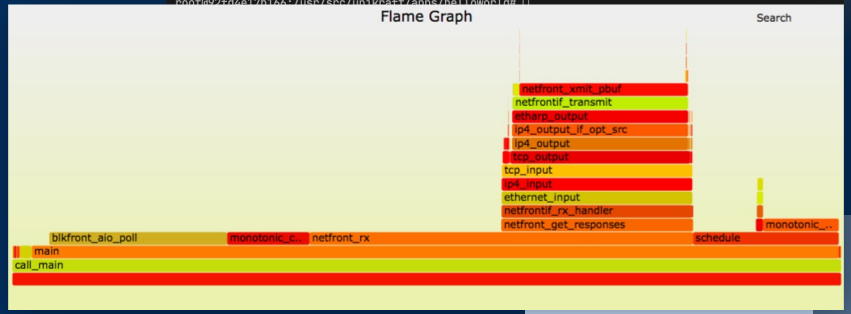
- ukdebug
  - Logging/Print system
  - Assertions
  - Tracepoints
  - GDB server
- uktest
  - Unit Testing
- ukstore
  - Directory of library getters and setters



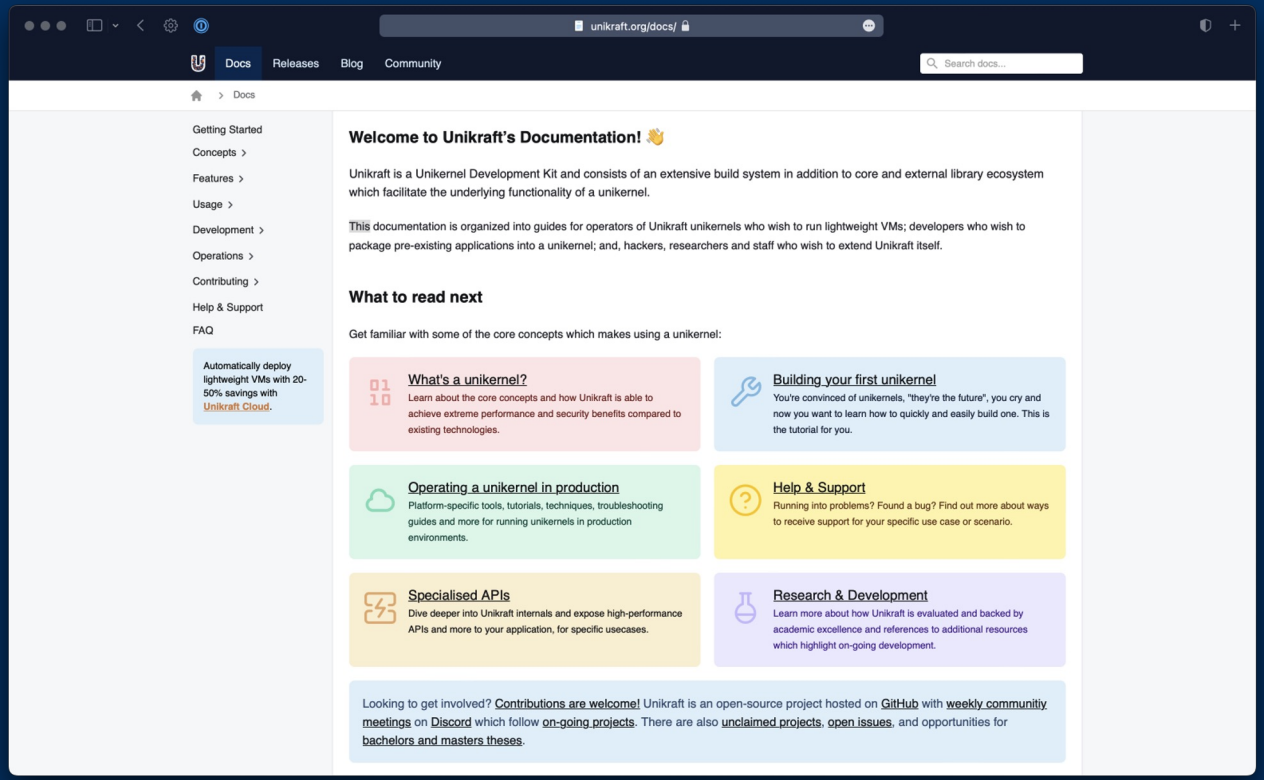
Monitoring & Debugging Features

- ukdebug
  - Logging/Print system
  - Assertions
  - Tracepoints
  - GDB server
- uktest
  - Unit Testing
- ukstore
  - Directory of library getters and setters
- uniprof (tool)
  - Performance analysis with stack snapshots


SeaBIOS (version rel-1.12.0-59-gc9ba5276e321-prebuilt.qemu.org)
Booting from ROM...
test: vfscore_mount_testsuite->vfscore_test_multimount
: expected `mkdir("/sys", S_IRWXU)` to be 0 but was 0 ..... FAILED
: in test_mount.c:42
: expected `mount("", "/sys", "ramfs", 0, NULL)` to be 0 but was 0 ..... FAILED
: in test_mount.c:43
: expected `mkdir("/dev", S_IRWXU)` to be 0 but was 0 ..... FAILED
: in test_mount.c:48
: expected `mount("", "/dev", "devfs", 0, NULL)` to be 0 but was 0 ..... FAILED
: in test_mount.c:49
: expected `mkdir("/tmp", S_IRWXU)` to be 0 but was 0 ..... FAILED
: in test_mount.c:51
: expected `mount("", "/tmp", "naivefs", 0, NULL)` to be 0 but was -1 ..... FAILED
: in test_mount.c:52
: expected `mount("", "/tmp", "naivetmpfs", 0, NULL)` to not be 0 but was -1 ..... PASSED
test: vfscore_stat_testsuite->vfscore_test_newfile
: expected `ret` to be 0 but was -1 ..... FAILED
: in test_stat.c:58
: expected `ret` to be 0 but was 0 ..... PASSED
: expected `fd` to be greater than 2 but was 3 ..... PASSED
: expected `write(fd, "hello\n", sizeof("hello\n"))` to be 7 but was 7 ..... PASSED
test: vfscore_stat_testsuite->vfscore_test_stat
: expected `rc` to be 0 but was 0 ..... PASSED
: expected `rc` to be 0 but was 0 ..... PASSED
: expected `rc` to be -1 but was -1 ..... PASSED
: expected `rc` to be 22 but was -22 ..... FAILED
: in test_stat.c:86
Powered by
Tethys 0.5.0~5b7f273-custom
Hello world!



Unikraft Documentation



Security

Security feature	Status	Targets
Stack Smashing Protection (SP)	Upstream	`ARCH_ARM_64 || ARCH_X86_64`
Undefined Behavior Sanitization (UBSAN)	Upstream	any
Rust internal libraries in Unikraft	Upstream	`ARCH_X86_64`
ARM Pointer authentication (PAuth)	Under review	`ARCH_ARM_64 || ARCH_ARM_32`
ARM Branch Target Identification (BTI)	Under review	`ARCH_ARM_64`
Kernel Address Sanitizer (KASAN)	Under review	`PLAT_KVM && ARCH_X86_64`
Position Independent Executables (PIE)	Under review	`PLAT_KVM && ARCH_X86_64`
True Random Number Generator	Under review	`ARCH_X86_64`
ARM Memory Tagging Extension (MTE)	Work-in-progress	ARM
Intel Control-flow Enforcement Technology (CET)	Planned	`ARCH_X86_64`
Shadow stack	Planned	any
`FORTIFY_SOURCE`	Planned	any
ARM Speculation Barrier (SB)	Planned	`ARCH_ARM_64`
Kernel Page Table Isolation (KPTI)	N/A	N/A
Supervisor Mode Access Prevention (SMAP)	N/A	N/A
Privileged Access Never (PAN)	N/A	N/A

Webpage Performance Test Result

DESKTOP | v100 | Cable | Frankfurt, Germany | More

View: Performance

Performance Optimization Overview

A detailed view of this site's asset optimization and related opportunities.

Re-run the test | Export Files

Optimization Summary

Quickly jump to the sections below:

- A+ Security score
- A First Byte Time
- A Keep-alive Enabled
- A Compress Transfer
- A Compress Images
- A Cache static content
- X Effective use of CDN

Unikraft Linux Foundation Project

unikraft.org

Docs Releases Blog Community

Search docs...

Unikraft

Quick Start

View on GitHub

What's a unikernel?

WHAT'S NEW Just released v0.8.0 (Enceladus) ›

Unikraft is a fast, secure and open-source Unikernel Development Kit

By tailoring the operating system, libraries and configuration to the particular needs of your application, it vastly reduces virtual machine and container image sizes to a few KBs, provides blazing performance, and drastically cuts down your software stack's attack surface.

- ✓ Blazing fast
- ✓ Developer-friendly
- ✓ Small footprint & green
- ✓ Cloud-native ready
- ✓ POSIX-compatible
- ✓ Fully modular
- ✓ Research-backed
- ✓ Feature-rich
- ✓ Production ready

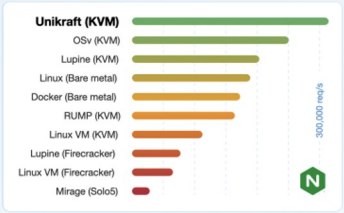
BLAZING FAST

Unikraft is faster than Linux

On Unikraft, NGINX is 166% faster than on Linux and 182% faster than on Docker

Unikraft outperforms well-configured custom Linux kernel images, even those with security mitigations turned off! Compared to other Unikernel Development Kits, library OSes and containers, Unikraft still comes out on top.

We benchmarked NGINX throughput in terms of requests per second compared to other unikernels, Linux and Docker; Unikraft achieves 182% performance improvement with respect to Docker.



Unikraft has been extensively and carefully benchmarked, read more about performance →

https://bit.ly/UnikraftDiscord

+ Romanian (yearly) Hackathon – September
+ UK Hackathon – Fall
+ Asian Hackathon – Spring
+ Your Hackathon here!!

Unikraft: Fast, Specialized Unikernels the Easy Way

Simon Kuenzer
NEC Laboratories Europe GmbH

Vlad-Andrei Bădoiu*
University Politehnica of Bucharest

Hugo Lefeuvre*
The University of Manchester

Sharan Santhanam*
NEC Laboratories Europe GmbH

Alexander Jung*
Lancaster University

Gaulthier Gain*
University of Liège

Cyril Soldani*
University of Liège

Costin Lupu
University Politehnica of Bucharest

Ștefan Teodorescu
University Politehnica of Bucharest

Costi Răducanu
University Politehnica of Bucharest

Cristian Banu
University Politehnica of Bucharest

Laurent Mathy
University of Liège

Răzvan Deaconescu
University Politehnica of Bucharest

Costin Raiciu
University Politehnica of Bucharest

Felipe Huici
NEC Laboratories Europe GmbH

Abstract

Unikernels are famous for providing excellent performance in terms of boot times, throughput and memory consumption, to name a few metrics. However, they are infamous for making it hard and extremely time consuming to extract such performance, and for needing significant engineering effort in order to port applications to them. We introduce Unikraft, a novel micro-library OS that (1) fully modularizes OS primitives so that it is easy to customize the unikernel

65], or providing efficient container environments [62, 76], to give some examples. Even in the hardware domain, and especially with the demise of Moore's law, manufacturers are increasingly leaning towards hardware specialization to achieve ever better performance; the machine learning field is a primary exponent of this [30, 32, 34].

In the virtualization domain, unikernels are the golden standard for specialization, showing impressive results in terms of throughput, memory consumption, and boot times,

https://github.com/unikraft/unikraft

https://unikraft.io

<info@unikraft.io>

@UnikraftSDK

The **Lightweight** Virtualization Company

Please
Star us on
GitHub!

https://github.com/unikraft/unikraft

https://unikraft.io

<info@unikraft.io>

@UnikraftSDK

The **Lightweight** Virtualization Company