The Lightweight Virtualization Company

KubeCon | CloudNativeCon

North America 2021

Deploying Unikernels in Production with Kubernetes

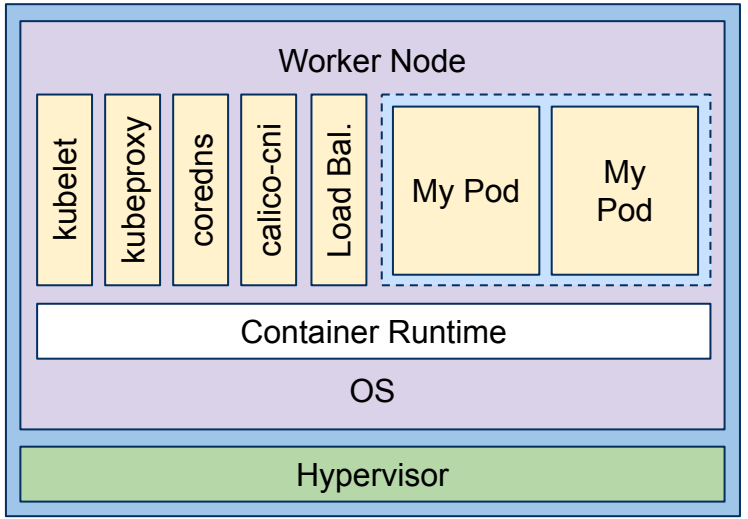
Alexander Jung <alexander.jung@unikraft.io>
Dominic Lindsay <d.lindsay4@lancs.ac.uk>
Laurentiu Barbulescu <lrbarbulescu@gmail.com>
Razvan Deaconescu <razvan.deaconescu@cs.pub.ro>
Felipe Huici <felipe.huici@unikraft.io>

Higher cluster utilization

&

Decrease Operational Expenditure

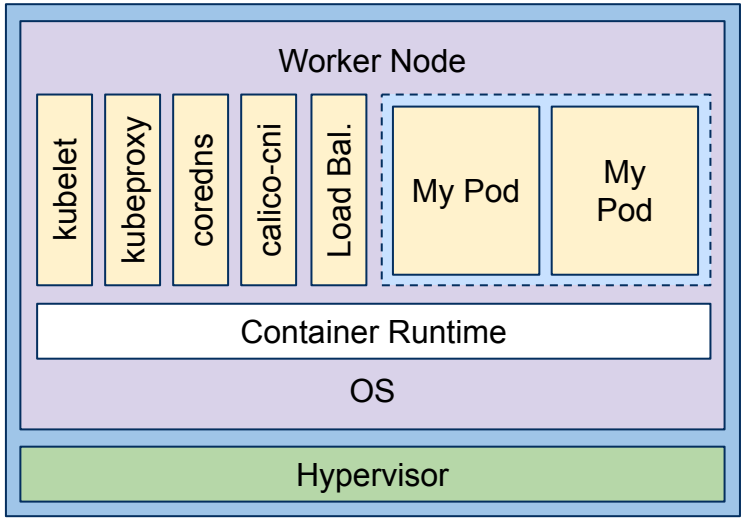
Anatomy of a “normal” K8s stack



Copyright © Unikraft UG.

Anatomy of a “normal” K8s stack

What’s the problem?



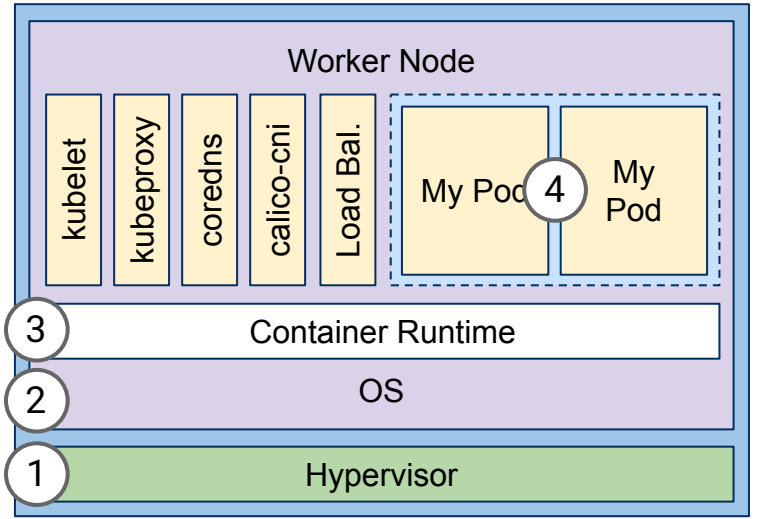
Copyright © Unikraft UG.

Anatomy of a “normal” K8s stack

What’s the problem?

- **Four degrees of virtualization**:
  Hypervisor, OS, Container Runtime...

  … and in some cases, the container offers internal isolation for processes!



Copyright © Unikraft UG.

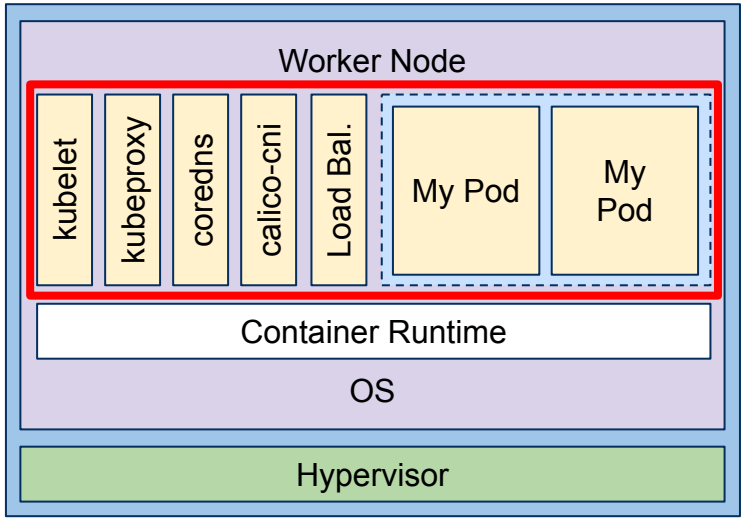
Anatomy of a “normal” K8s stack

What’s the problem?

- **Four degrees of virtualization**:
  Hypervisor, OS, Container Runtime...

  … and in some cases, the container offers internal isolation for processes!

- **Kube-system** pods and *My Pod*s are separated only by namespaces and use the same kernel!



Copyright © Unikraft UG.

Reducing the OS with specialization

Existing k8s-specific OSes “strip away” unnecessary system libs, and tailor the OS for running kubernetes...

Copyright © Unikraft UG.

Reducing the OS with specialization

Existing k8s-specific OSes “strip away” unnecessary system libs, and tailor the OS for running kubernetes...

… But they still use a monolithic-kernel (Linux)

Copyright © Unikraft UG.

Reducing the OS with specialization

Existing k8s-specific OSes “strip away” unnecessary system libs, and tailor the OS for running kubernetes...

… But they still use a monolithic-kernel (Linux)

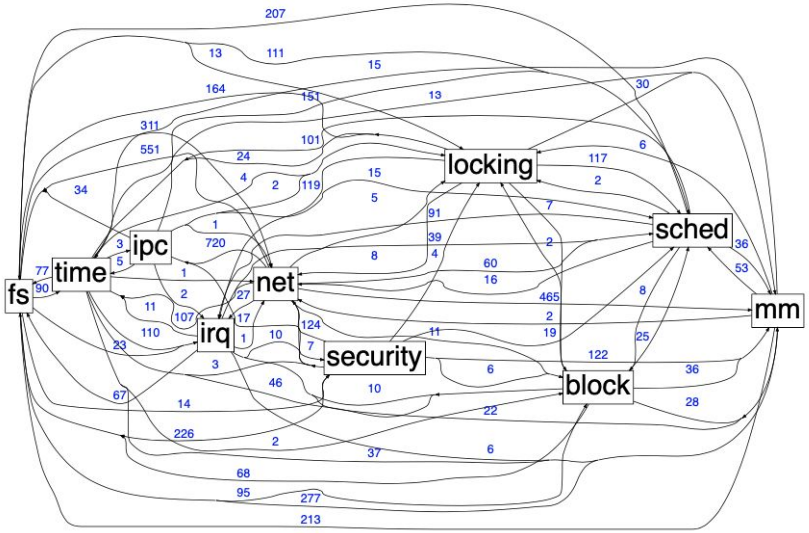
... And they still run on a hypervisor

Copyright © Unikraft UG.

Could we reduce Linux further?

Copyright © Unikraft UG.

Could we reduce Linux further?



Linux kernel components have strong inter-dependencies, making it difficult to remove or replace them.

Copyright © Unikraft UG.

Higher cluster utilization

&

Decrease Operational Expenditure

andreessen horowitz It's time to build

Portfolio Team Focus Areas Content About Jobs

The Cost of Cloud, a Trillion Dollar Paradox

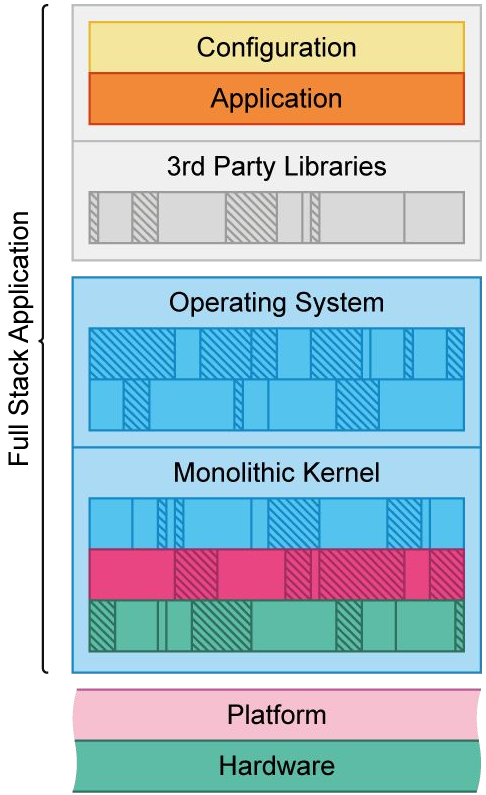
by Sarah Wang and Martin Casado

cloud computing • enterprise & SaaS • networking • growth (late stage venture) • metrics • cloud infrastructure • trends 2021

There is no doubt that the cloud is one of the most significant platform shifts in the history of computing. Not only has cloud already impacted hundreds of billions of dollars of IT spend, it's still in early innings and growing rapidly on a base of over $100B of annual public cloud spend. This shift is driven by an incredibly powerful value proposition — infrastructure available immediately, at exactly the scale needed by the business — driving efficiencies both in operations and economics. The cloud also helps cultivate innovation as company resources are freed up to focus on new products and growth.

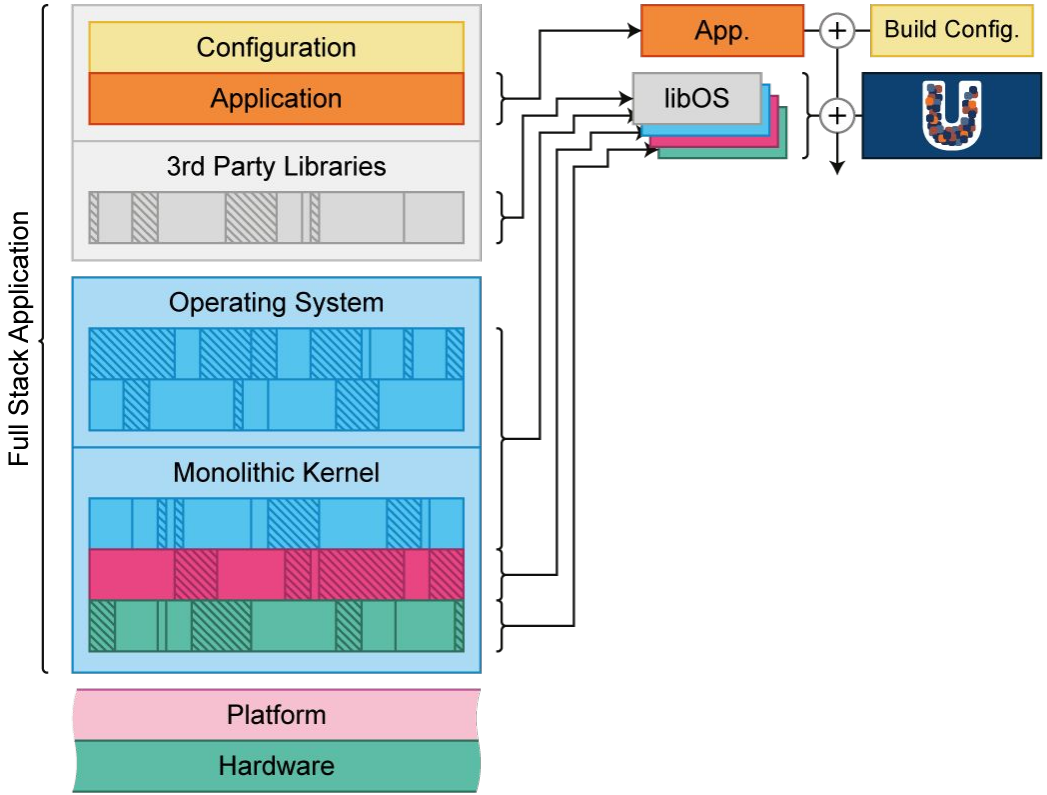
Introducing Unikernels

The Unikernel Model



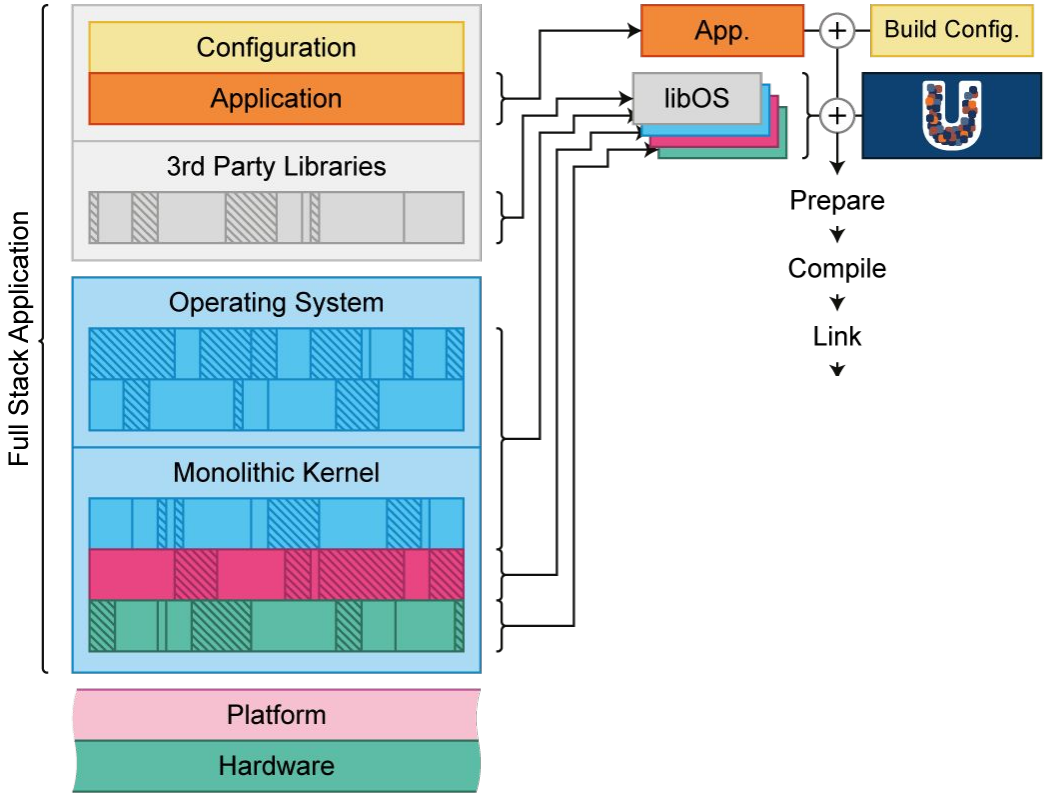
Copyright © Unikraft UG.

The Unikernel Model



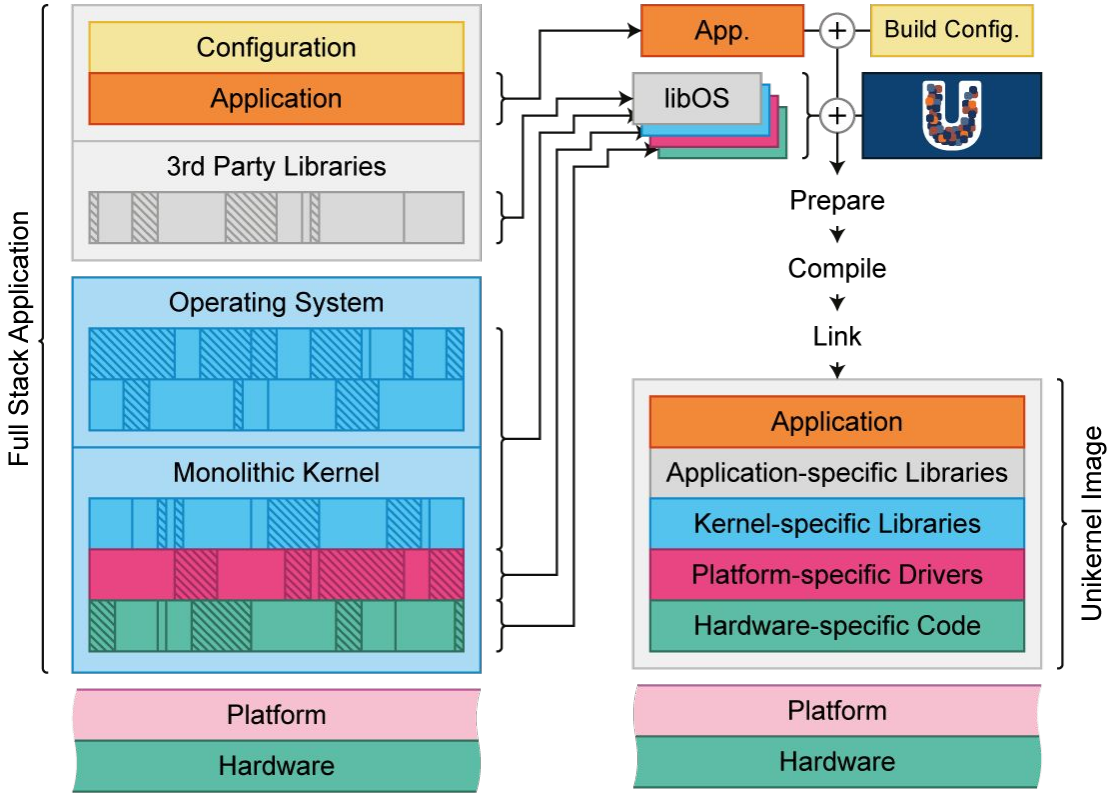
Copyright © Unikraft UG.

The Unikernel Model



Copyright © Unikraft UG.

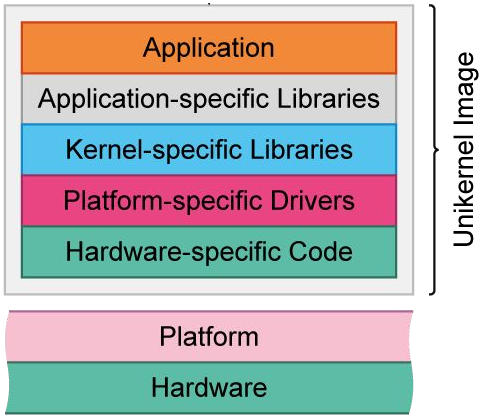
The Unikernel Model



Copyright © Unikraft UG.

The Unikernel Model

Unikernels:

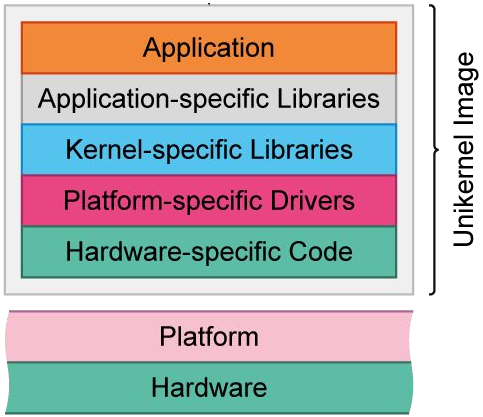


Copyright © Unikraft UG.

The Unikernel Model

Unikernels:

→ Compile-time specialization strategy;



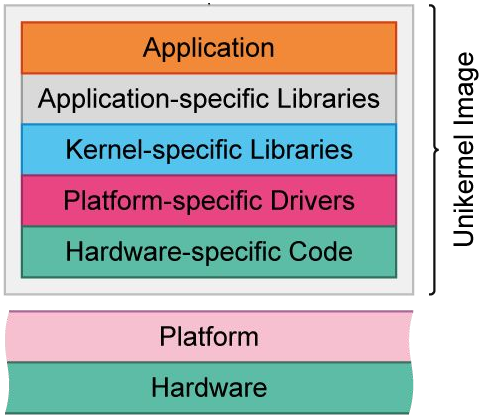
Copyright © Unikraft UG.

The Unikernel Model

Unikernels:

→ Compile-time specialization strategy;

→ Lightweight Virtual Machines;



Copyright © Unikraft UG.

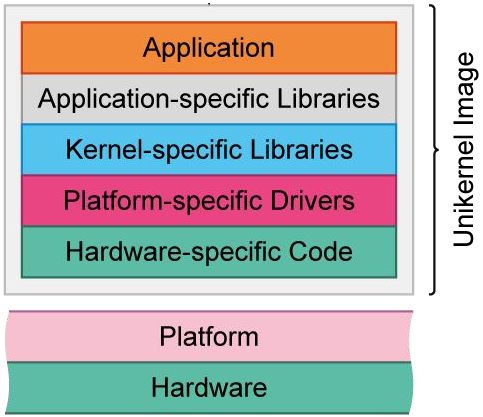
The Unikernel Model

Unikernels:

→ Compile-time specialization strategy;

→ Lightweight Virtual Machines;

→ Single, Sealed Address-Space;



Copyright © Unikraft UG.

The Unikernel Model

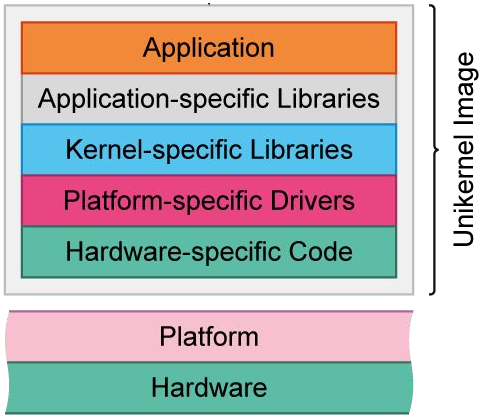
Unikernels:

→ Compile-time specialization strategy;

→ Lightweight Virtual Machines;

→ Single, Sealed Address-Space;

→ No costly syscalls;

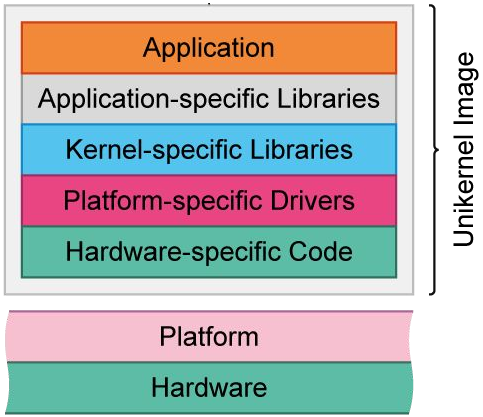


Copyright © Unikraft UG.

The Unikernel Model

Unikernels:

- → Compile-time specialization strategy;
- → Lightweight Virtual Machines;
- → Single, Sealed Address-Space;
- → No costly syscalls;
- → Only necessary functionality for application to run;

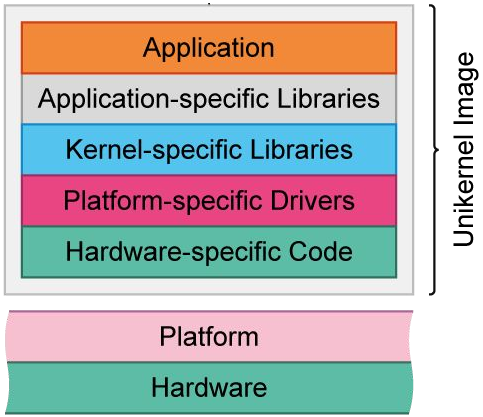


Copyright © Unikraft UG.

The Unikernel Model

Unikernels:

- → Compile-time specialization strategy;
- → Lightweight Virtual Machines;
- → Single, Sealed Address-Space;
- → No costly syscalls;
- → Only necessary functionality for application to run;
- → No daemons, system libs, SSH;

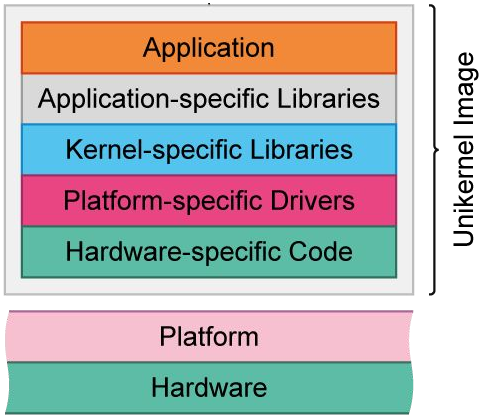


Copyright © Unikraft UG.

The Unikernel Model

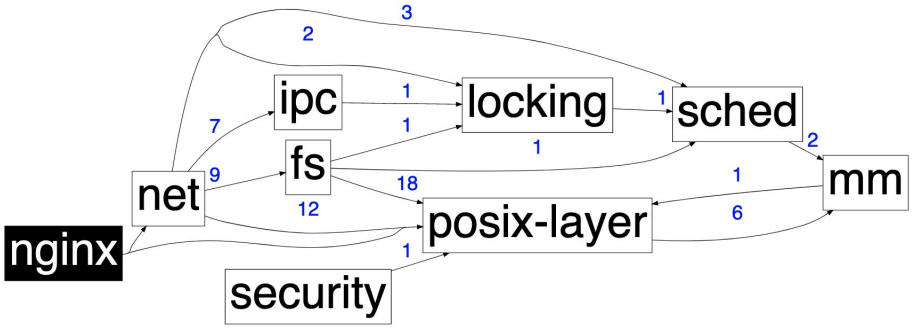
Unikernels:

- → Compile-time specialization strategy;
- → Lightweight Virtual Machines;
- → Single, Sealed Address-Space;
- → No costly syscalls;
- → Only necessary functionality for application to run;
- → No daemons, system libs, SSH;
- → Platform-/Hardware-specific.

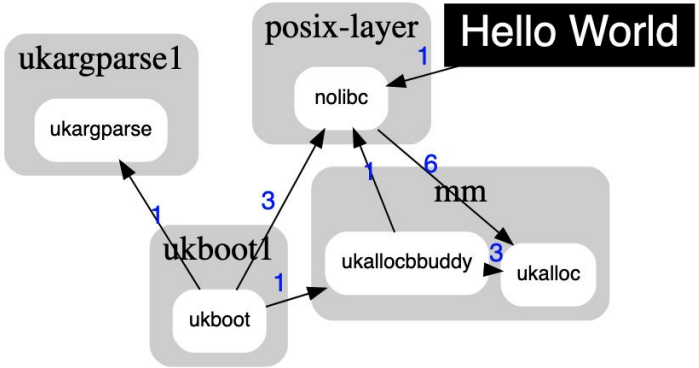


Copyright © Unikraft UG.

We can reduce complexity with Unikraft



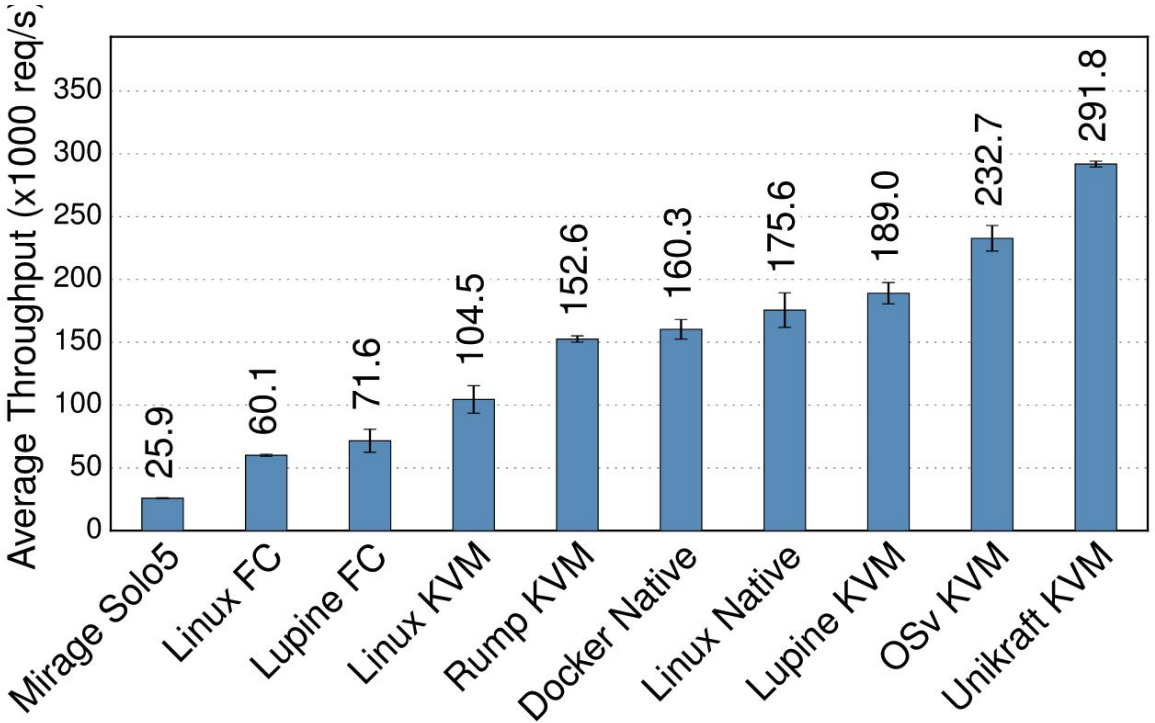
NGINX with Unikraft shows fewer real dependencies...



...Because Unikraft is built to suit the application's needs and nothing more.

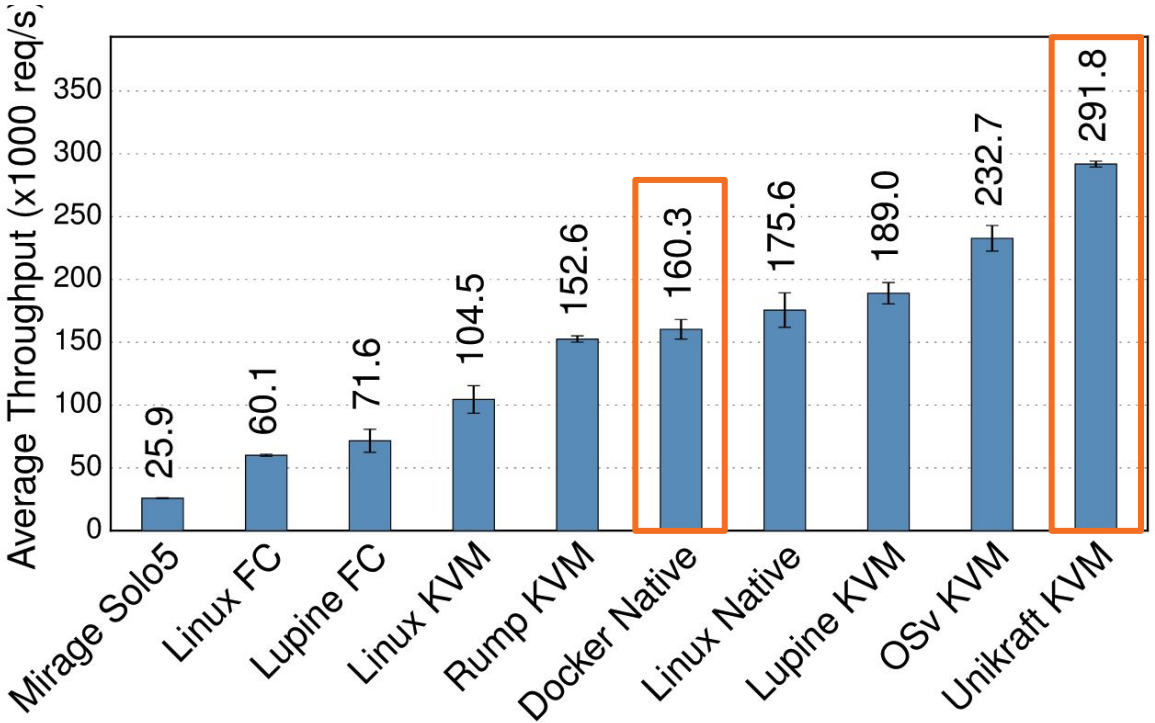
Copyright © Unikraft UG.

Unikernels offer better performance.



Copyright © Unikraft UG.

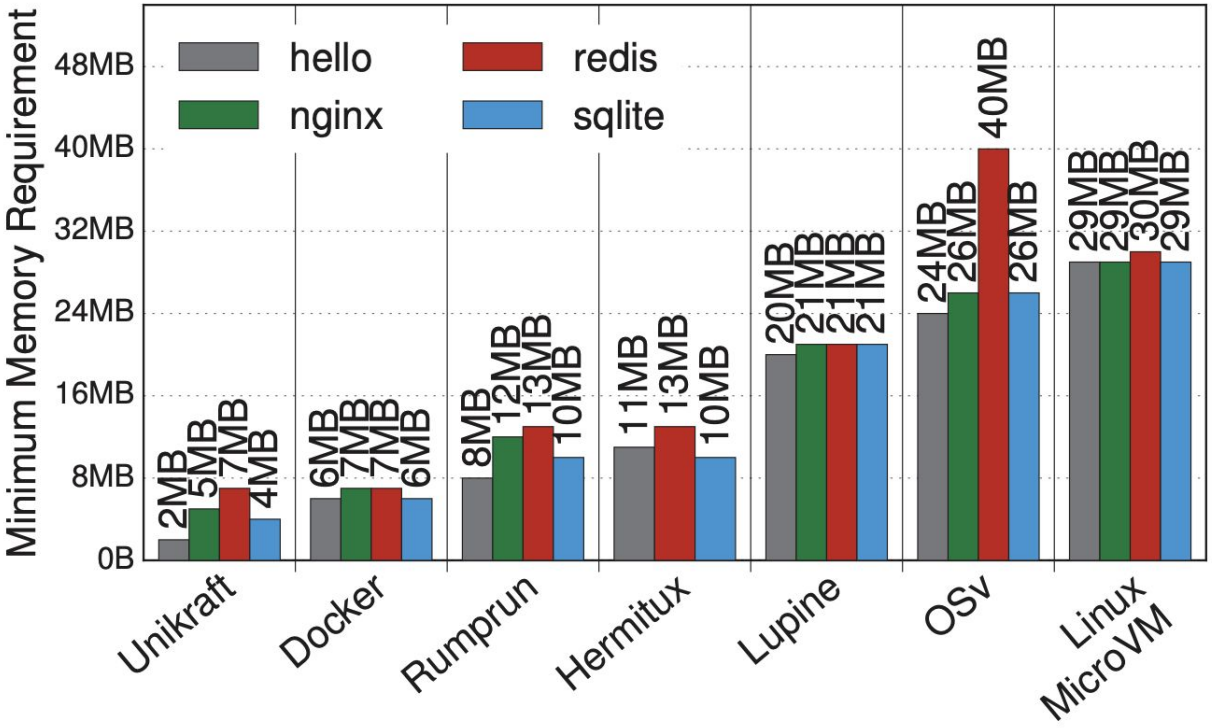
Unikernels offer better performance.



82% increase
in performance

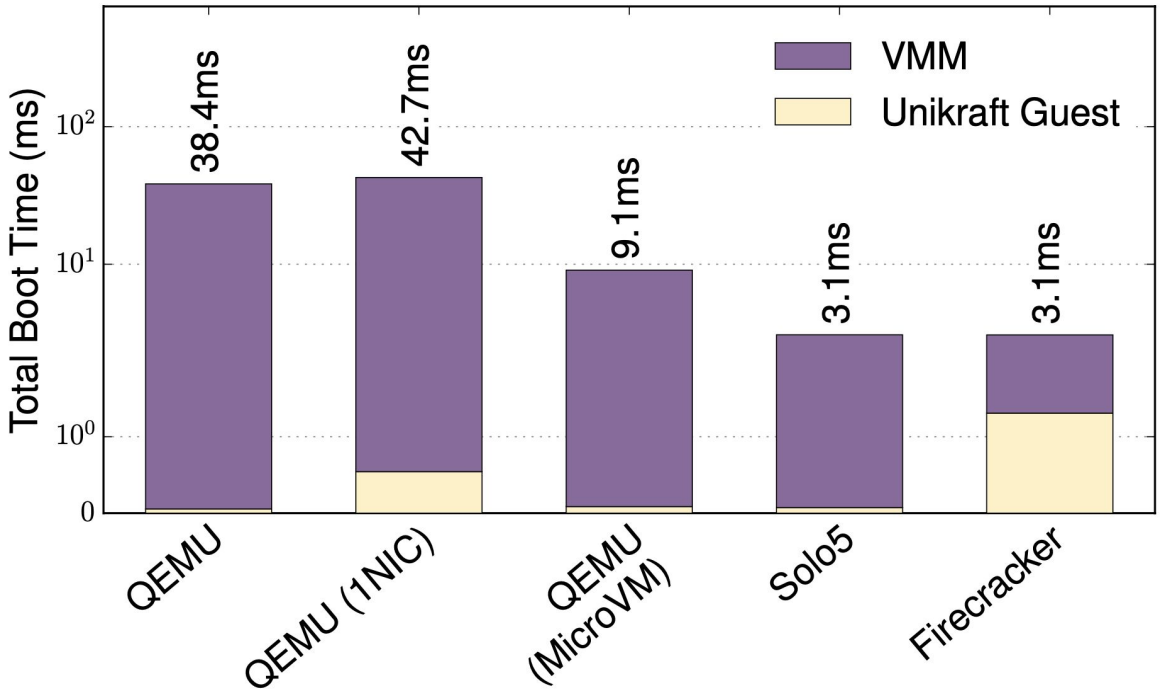
Copyright © Unikraft UG.

Unikernels offer better performance.



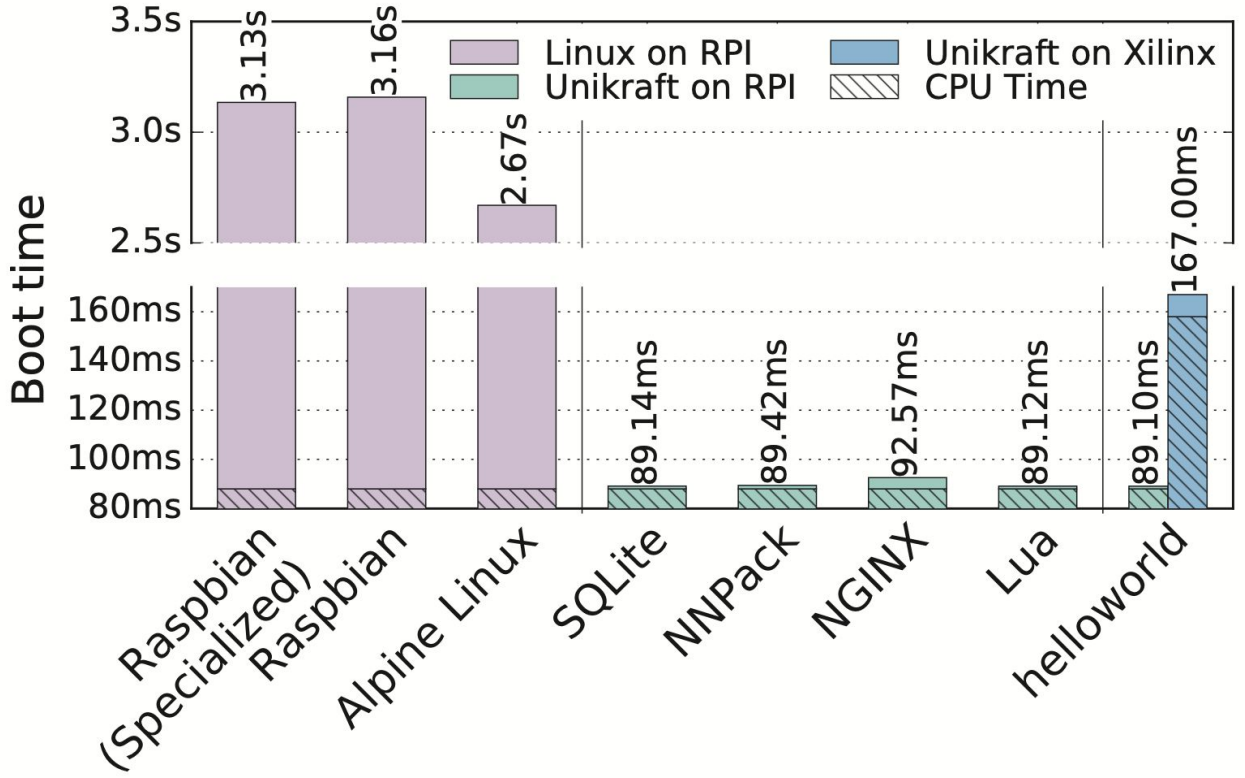
Copyright © Unikraft UG.

Unikernels offer better performance.



Copyright © Unikraft UG.

Unikernels offer better performance.



Copyright © Unikraft UG.

Unikernels offer better performance.

Platform	Routine call	#Cycles	nsecs
Linux/KVM	System call	222.0	61.67
	System call (no mitigations)	154.0	42.78
Unikraft/KVM	System call	84.0	23.33
Both	Function call	4.0	1.11

Copyright © Unikraft UG.

Unikernels offer better transport.

Image	Size
docker.io/nginx:1.15.6	
unikraft.io/nginx:1.15.6	

Copyright © Unikraft UG.

Unikernels offer better transport.

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

Copyright © Unikraft UG.

Unikernels offer better transport.

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

Copyright © Unikraft UG.

Unikernels offer better security.

- Reduced attack surface:
  - → No shell;
  - → No users;
  - → No background processes

Copyright © Unikraft UG.

Unikernels offer better security.

- Reduced attack surface:
  - → No shell;
  - → No users;
  - → No background processes
- Reduced memory space & ASLR
  - → Lower attack surface

Copyright © Unikraft UG.

Unikernels offer better security.

- Reduced attack surface:
  - → No shell;
  - → No users;
  - → No background processes
- Reduced memory space & ASLR
  - → Lower attack surface
- Hardware-based isolation
  - → Lowest-level virtualization used

Copyright © Unikraft UG.

Unikernels offer better security.

- Reduced attack surface:
  - → No shell;
  - → No users;
  - → No background processes
- Reduced memory space & ASLR
  - → Lower attack surface
- Hardware-based isolation
  - → Lowest-level virtualization used
- Inner-VM memory protection with hardware acceleration (*on-going work*)

Copyright © Unikraft UG.

Our Goal

1. We envision dynamic workloads of unikernels scheduled via Kubernetes

Copyright © Unikraft UG.

Our Goal

1. We envision dynamic workloads of unikernels scheduled via Kubernetes

2. Little-to-no modifications to the host: be “pluginizable”

Copyright © Unikraft UG.

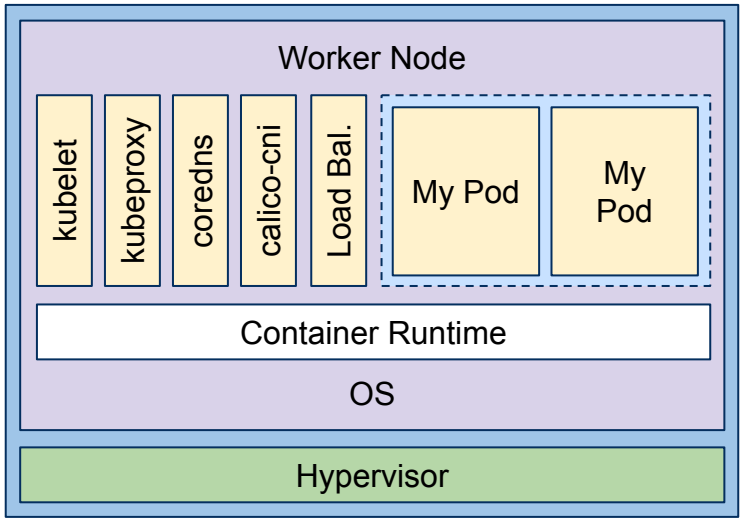
Interfacing VMs with K8s

- But it is possible already to interface with VMs!

Copyright © Unikraft UG.

Interfacing VMs with K8s

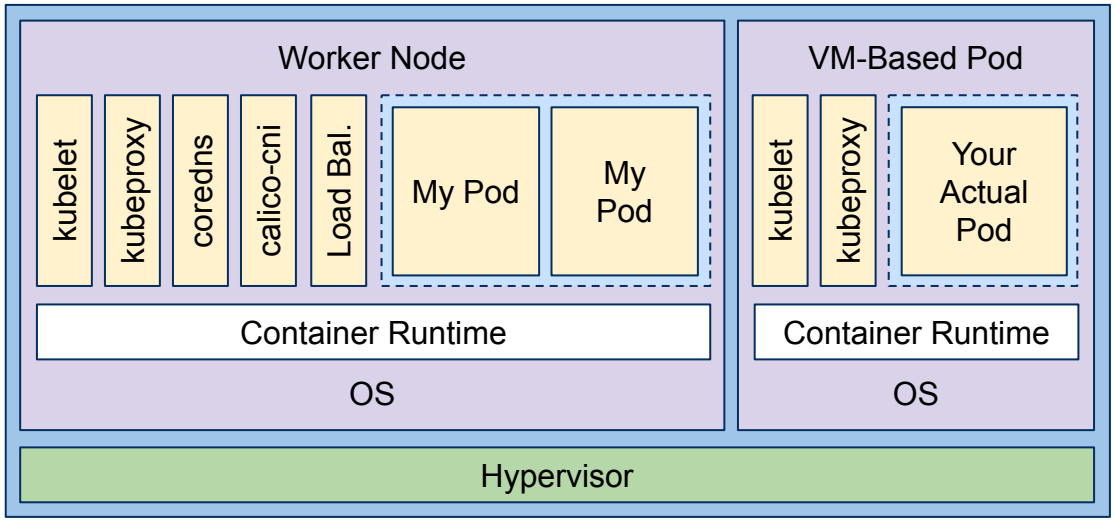
- But it is possible already to interface with VMs!



Copyright © Unikraft UG.

Interfacing VMs with K8s

- But it is possible already to interface with VMs!



Copyright © Unikraft UG.

Interfacing VMs with K8s

- Kubernetes talks to a container runtime engine via Container Runtime Interface (CRI), e.g. `containerd`

Copyright © Unikraft UG.

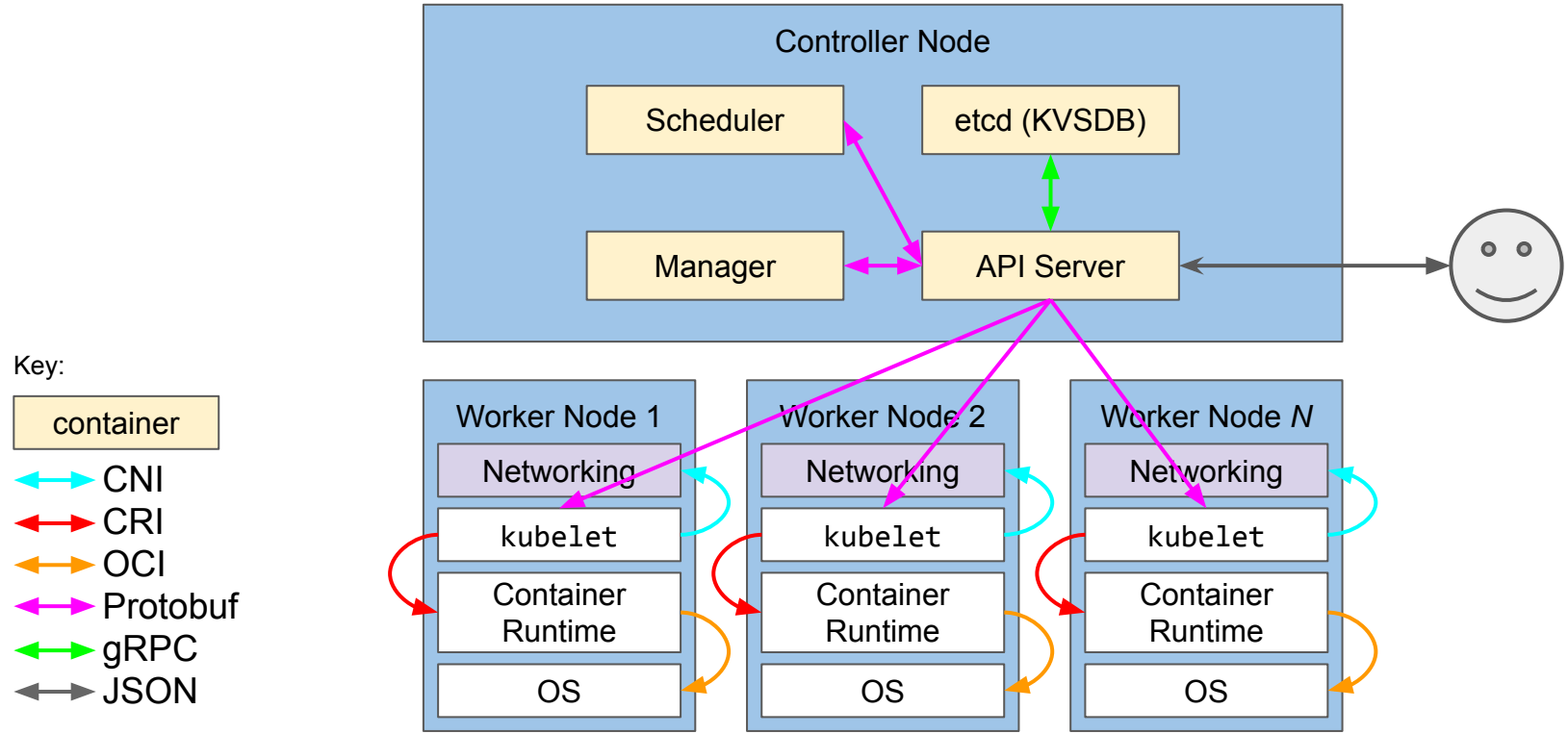
Interfacing VMs with K8s

- Kubernetes talks to a container runtime engine via Container Runtime Interface (CRI), e.g. `containerd`

- The container runtime engine speaks with an OCI Runtime/Image Specification compliant program, e.g. `runc`

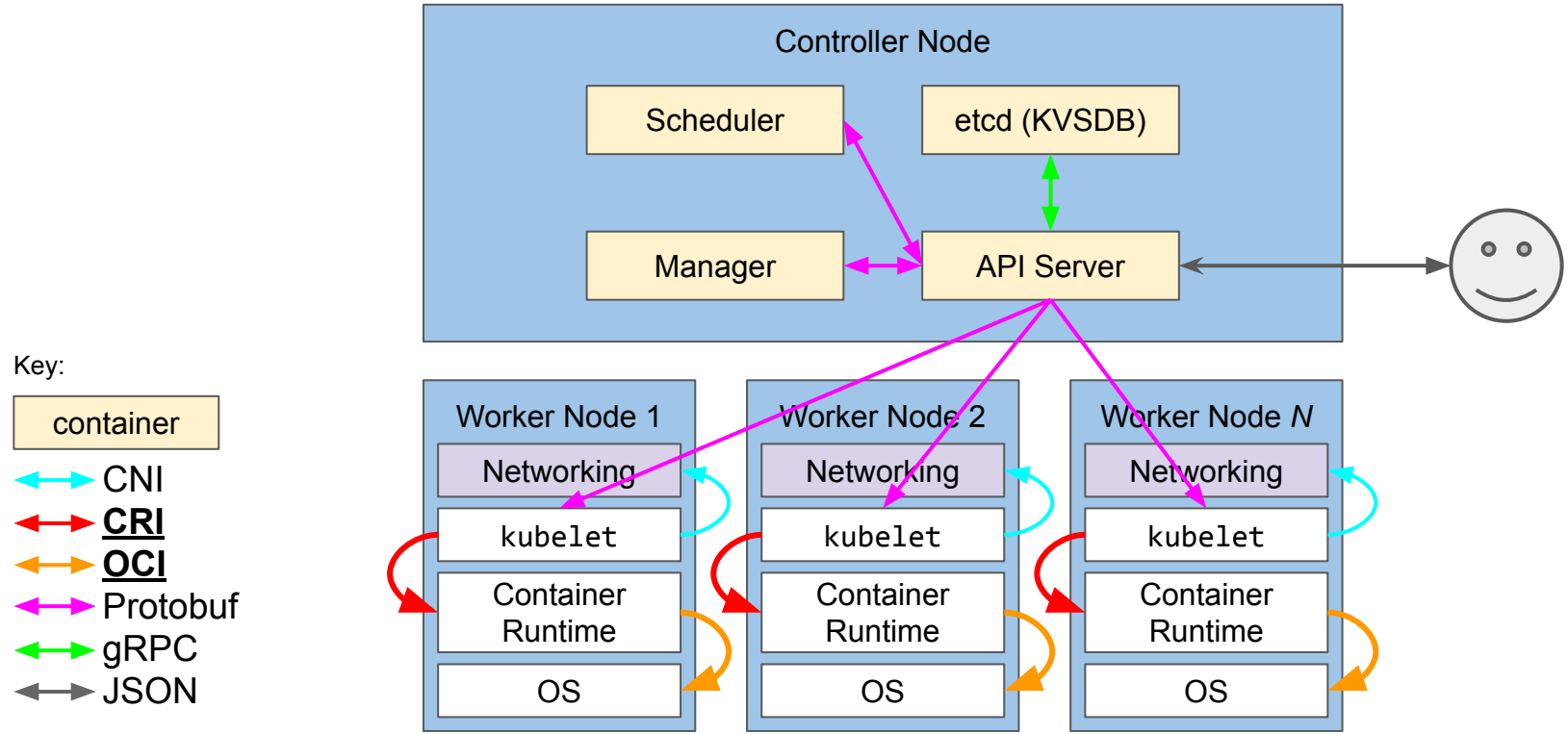
Copyright © Unikraft UG.

Interfacing VMs with K8s



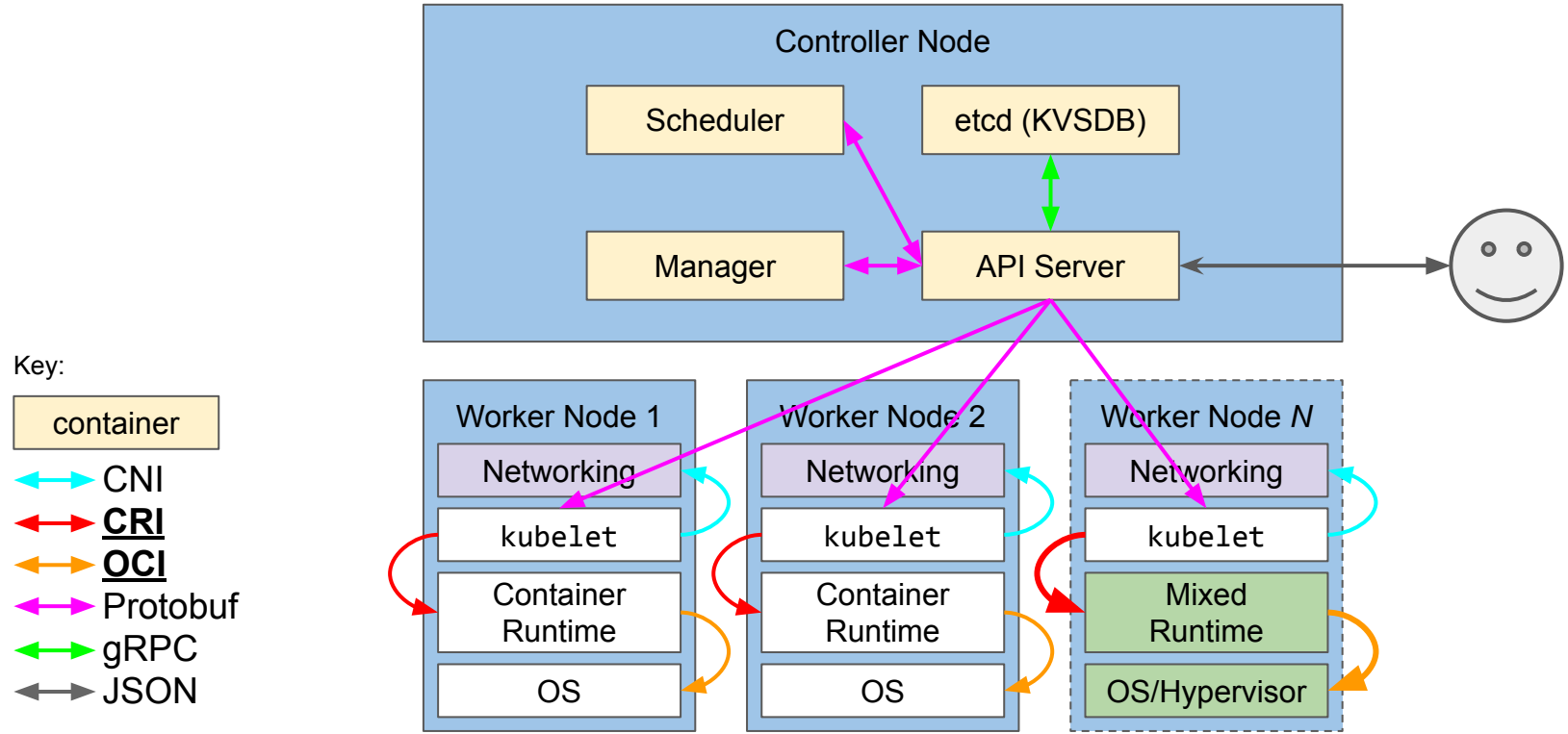
Copyright © Unikraft UG.

Interfacing VMs with K8s



Copyright © Unikraft UG.

Interfacing VMs with K8s

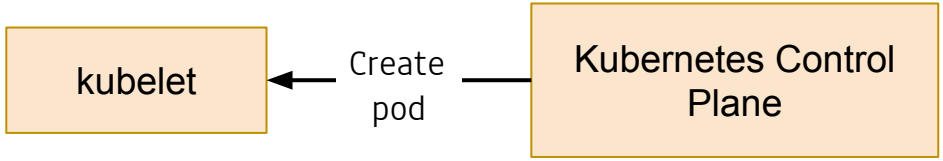


Copyright © Unikraft UG.

Interfacing VMs with K8s

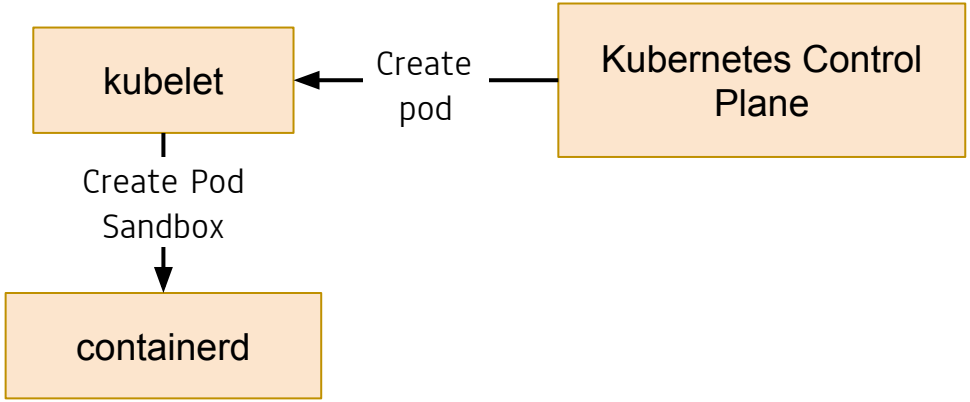
Copyright © Unikraft UG.

Interfacing VMs with K8s



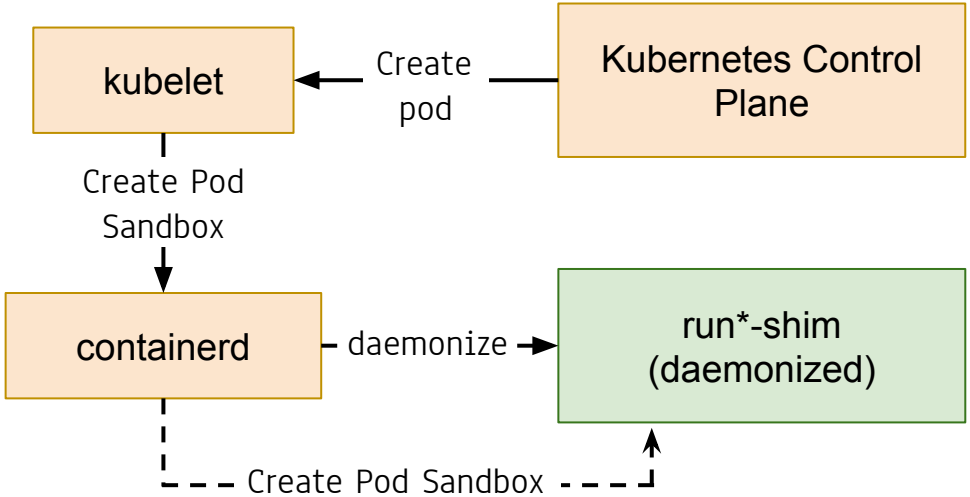
Copyright © Unikraft UG.

Interfacing VMs with K8s



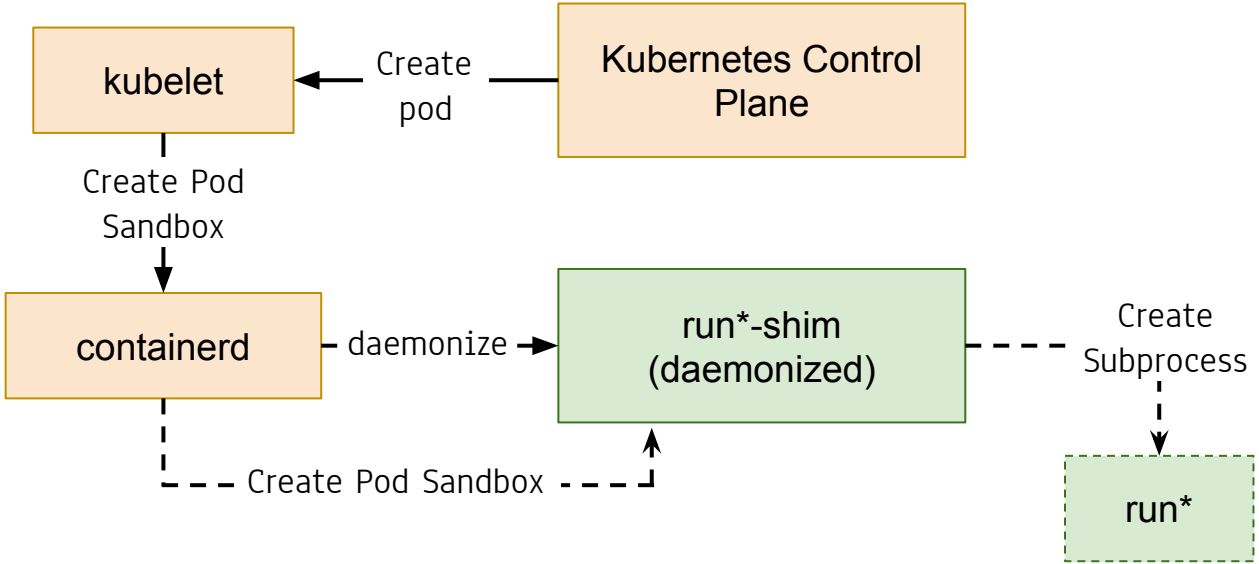
Copyright © Unikraft UG.

Interfacing VMs with K8s



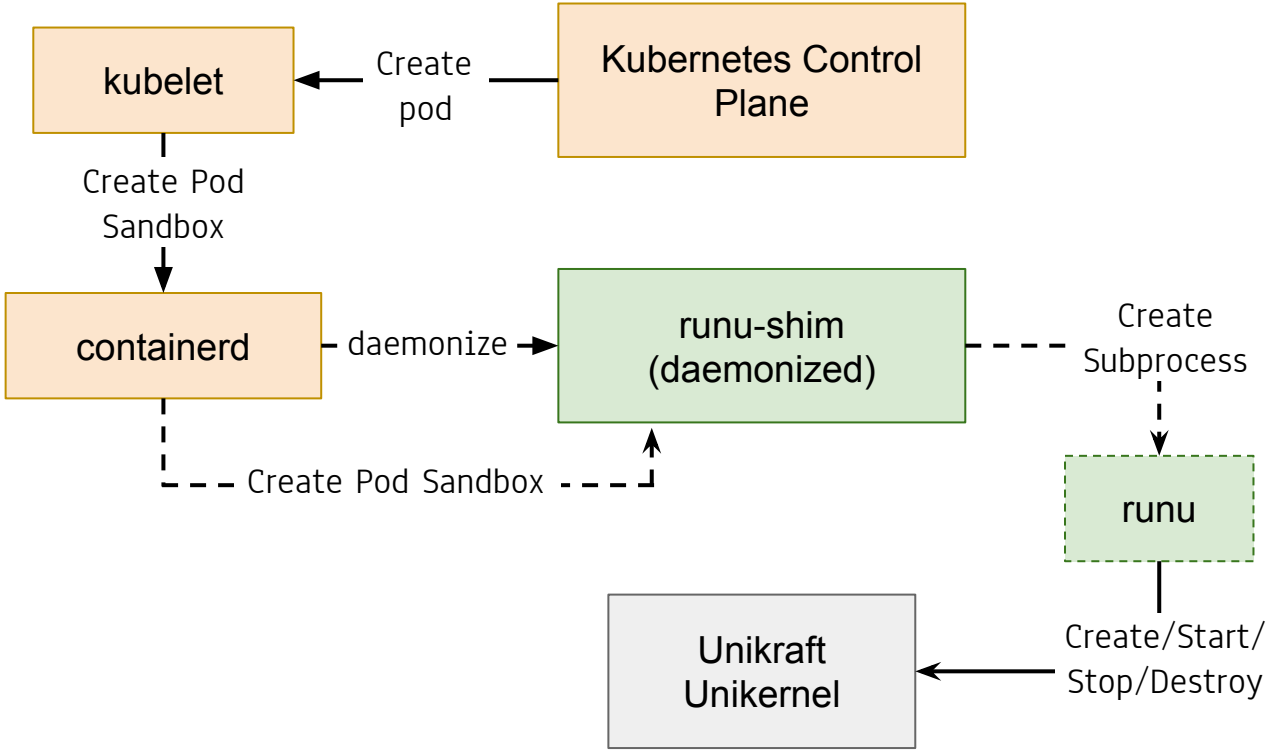
Copyright © Unikraft UG.

Interfacing VMs with K8s



Copyright © Unikraft UG.

Interfacing VMs with K8s



Copyright © Unikraft UG.

Extending the Container Runtime

- `containerd` can very easily introduce new runtimes:

```
/etc/containerd/config.toml
```

Copyright © Unikraft UG.

Extending the Container Runtime

- `containerd` can very easily introduce new runtimes:

`/etc/containerd/config.toml`

```
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runu]
  runtime_type = "io.containerd.runtime.runu.v2"
  runtime_engine = ""
  runtime_root = ""
  privileged_without_host_devices = false
  base_runtime_spec = ""
    [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runu.options]
      BinaryName = "/usr/local/bin/runu"
```

Copyright © Unikraft UG.

Extending the Container Runtime

- `containerd` can very easily introduce new runtimes:

`/etc/containerd/config.toml`

```
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runu]
  runtime_type = "io.containerd.runtime.runu.v2"
  runtime_engine = ""
  runtime_root = ""
  privileged_without_host_devices = false
  base_runtime_spec = ""
    [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runu.options]
      BinaryName = "/usr/local/bin/runu"
```

Copyright © Unikraft UG.

Extending the Container Runtime

- `containerd` can very easily introduce new runtimes:

`/etc/containerd/config.toml`

```
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runu]
  runtime_type = "io.containerd.runtime.runu.v2"
  runtime_engine = ""
  runtime_root = ""
  privileged_without_host_devices = false
  base_runtime_spec = ""
    [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runu.options]
      BinaryName = "/usr/local/bin/runu"
```

Copyright © Unikraft UG.

runu: Run Unikernels!

```
runu, an OCI compatible unikernel runtime interface.

Usage:
  runu [command]

Available Commands:
  create      Create a unikernel with given ID and bundle.
  delete      Delete a unikernel with a given ID.
  help        Help about any command.
  kill        Kill a unikernel with a given ID.
  spec        Create a runtime-spec from a unikernel-spec
  start       Start a unikernel with a given ID.
  state       Query state of a unikernel.

Flags:
  -h, --help   help for runu

Use "runu [command] --help" for more information about a command.
```

- To be OCI compliant, provide a binary with the following subcommands and args:

  → `create <id> <bundle>`
  → `delete <id>`
  → `kill <id> [signal]`
  → `spec ...`
  → `start <id>`
  → `state <id>`

Copyright © Unikraft UG.

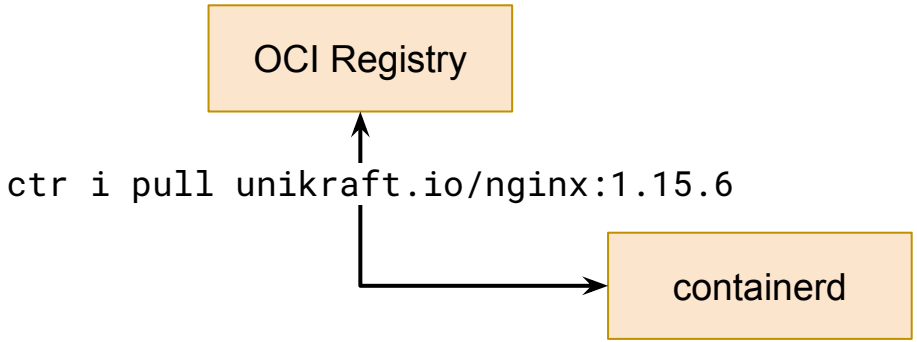
A unikernel in OCI clothing

- `containerd` has an image service, which pulls the OCI image

Copyright © Unikraft UG.

A unikernel in OCI clothing

- `containerd` has an image service, which pulls the OCI image



Copyright © Unikraft UG.

A unikernel in OCI clothing

Copyright © Unikraft UG.

A unikernel in OCI clothing

The OCI image contains mandatory manifest (`index.json`) and a number of “layers”, which include: filesystem (initramfs); mounted filesystem (NGINX config).

`unikraft.io/nginx:1.15.6`

```
.
├── blobs/
│   └── sha256/
│       ├── 872ba4b589c3c271e1b91d7de9b94697eb2c638a6b189ab4d63892f86456a9c1
│       ├── a2eb4c1b70ee5219c184005b3243f05d254dbd3c2653b7745076db5a7bbd7e42
│       └── ...
├── index.json
└── oci-layout
```

Copyright © Unikraft UG.

A unikernel in OCI clothing

The OCI image contains mandatory manifest (`index.json`) and a number of “layers”, which include: filesystem (initramfs); mounted filesystem (NGINX config).

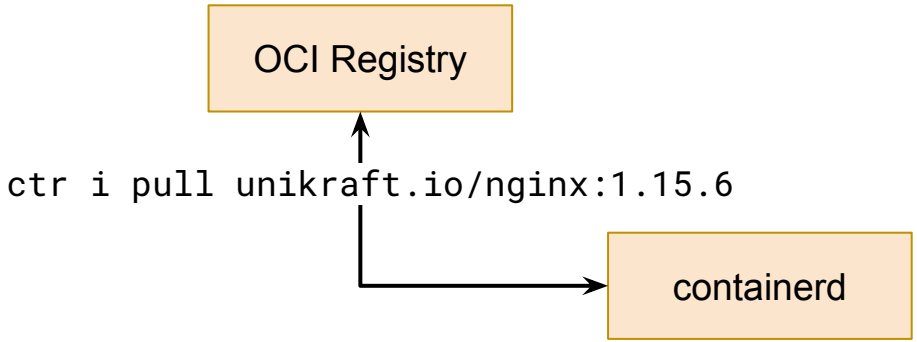
`unikraft.io/nginx:1.15.6`

```
blobs/sha256/
├── 872ba4b589c3c271e1b91d7de9b94697eb2c638a6b189ab4d63892f86456a9c1/
│   ├── nginx_kvm-x86_64
│   └── initramfs.cpio
└── a2eb4c1b70ee5219c184005b3243f05d254dbd3c2653b7745076db5a7bbd7e42/
    └── config.json
```

Copyright © Unikraft UG.

A unikernel in OCI clothing

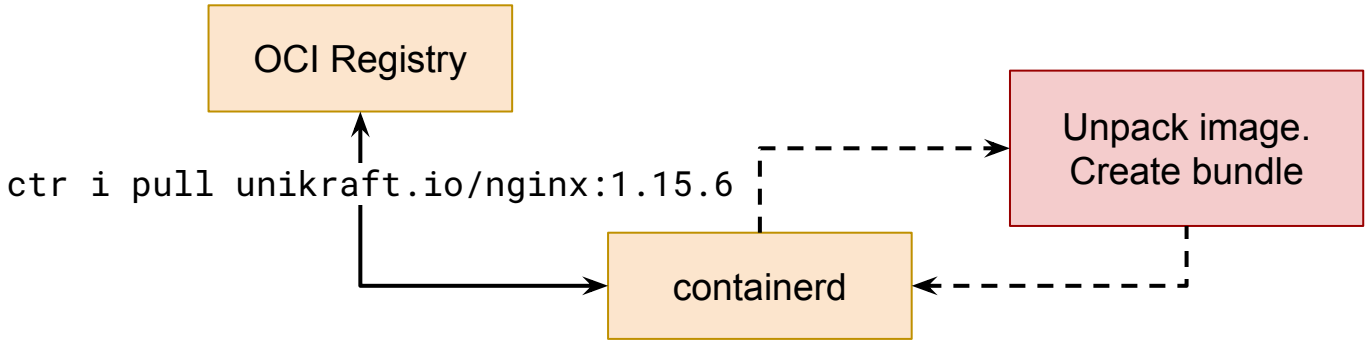
- `containerd` has an image service, which pulls the OCI image



Copyright © Unikraft UG.

A unikernel in OCI clothing

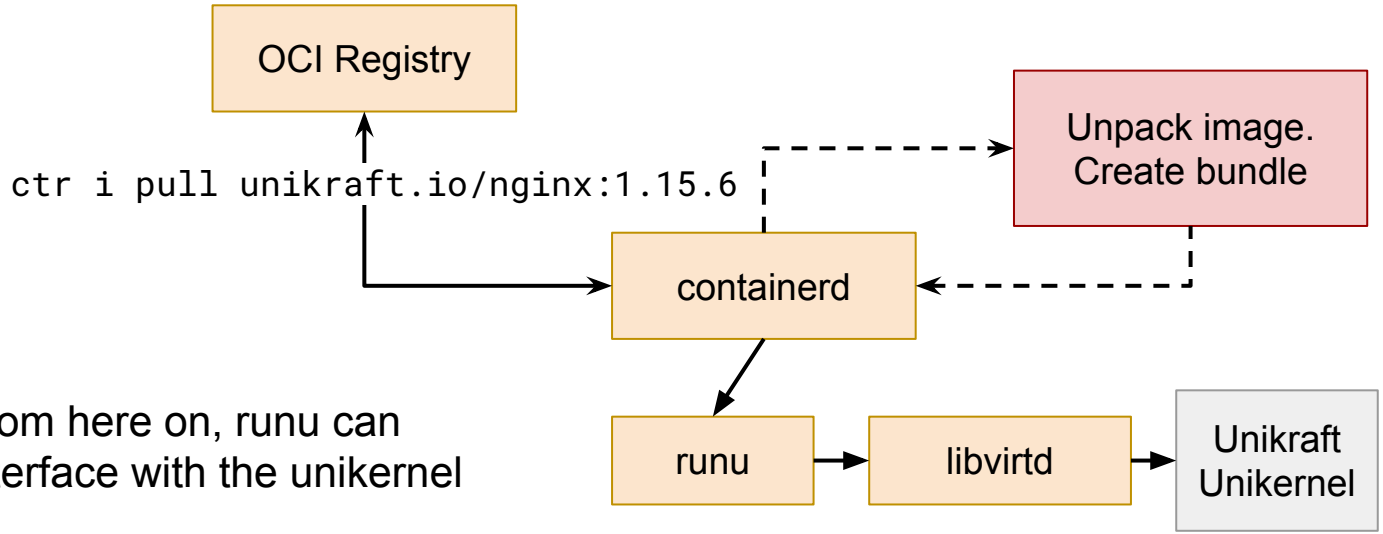
- `containerd` has an image service, which pulls the OCI image
- This image is passed to the runtime engine as “bundle” (image + spec):



Copyright © Unikraft UG.

A unikernel in OCI clothing

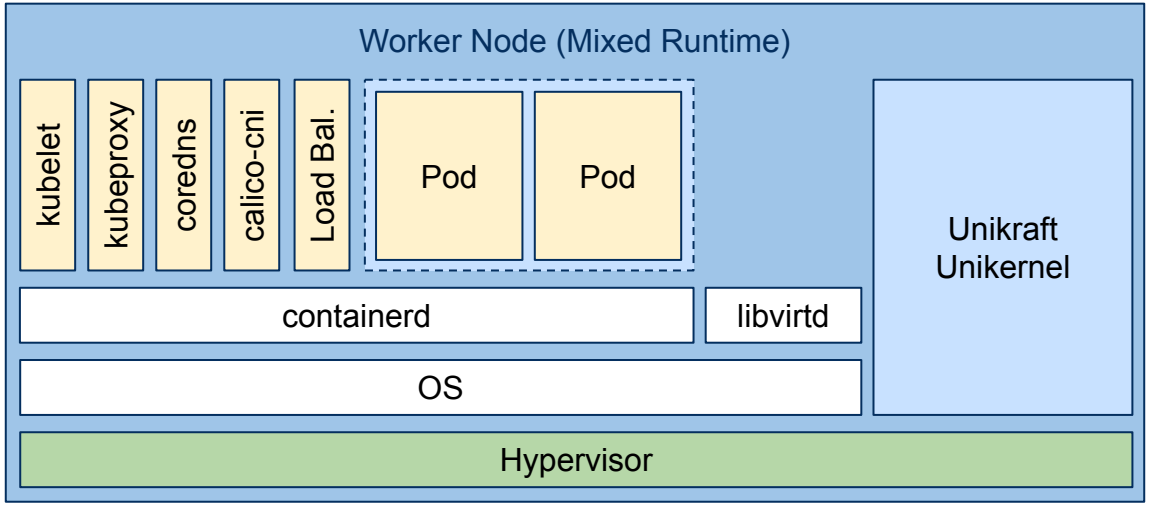
- `containerd` has an image service, which pulls the OCI image
- This image is passed to the runtime engine as “bundle” (image + spec):



Copyright © Unikraft UG.

Mixed Container/Unikernel Runtime

- OS must be able to communicate with Hypervisor (libvirtd), (changed required)
- runu can be deployed to Kubernetes (no change to host required)



Copyright © Unikraft UG.

Demo

Future work

- Improve OCI image contents for better libvirt integration… *lots of metadata*!
- Experimenting with new schedulers for co-location of unikernels;
- OCI registry unikernel “image matrix” for diverse, heterogeneous clusters

Copyright © Unikraft UG.

THANKS !

The Lightweight Virtualization Company

@UnikraftSDK

https://unikraft.io