The -ization of Containerization

John Logan – Container Team

container tool

Containerization

What are containers?

Containerization principles

Extending Container

Building on Containerization

Adoption of Swift

Open Source Community

What are containers?

Containerization principles

Extending Container

Building on Containerization

Adoption of Swift

Open Source Community

Runtime properties

- Provide isolation from the host

Runtime properties

- Provide isolation from the host
- Provide isolation from other workloads

Runtime properties

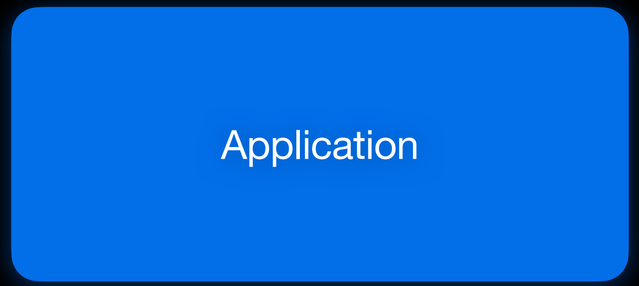
- Provide isolation from the host
- Provide isolation from other workloads
- Development environments

Runtime properties

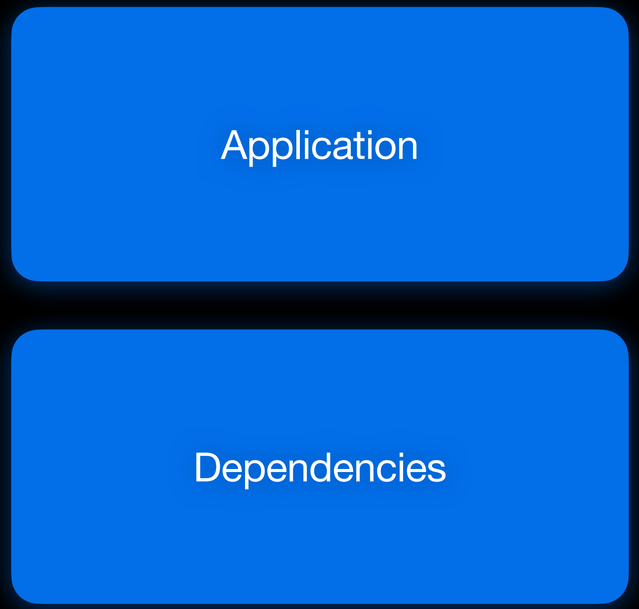
- Provide isolation from the host
- Provide isolation from other workloads
- Development environments

> Lightweight and fast

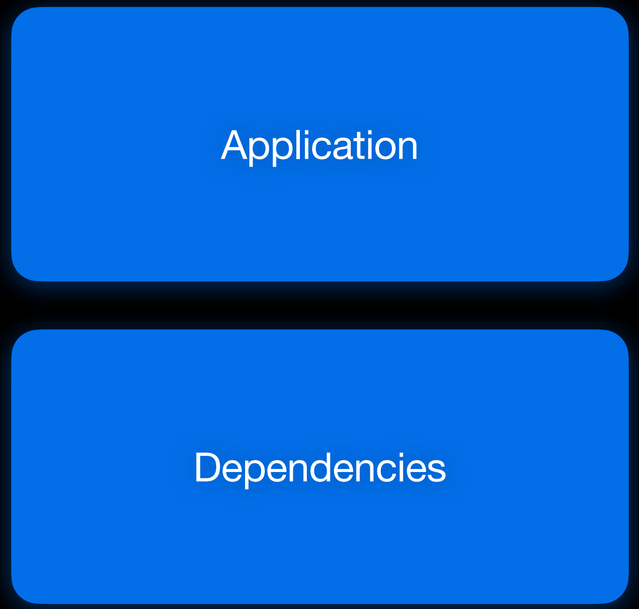
Packaging properties



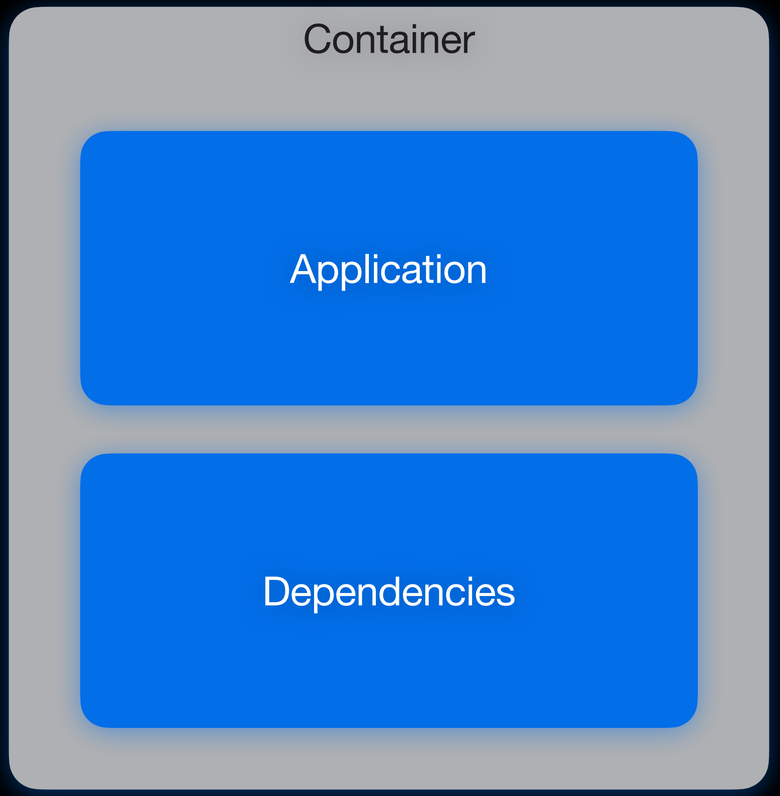
Packaging properties



Packaging properties



Packaging properties



Demonstration

What are containers?

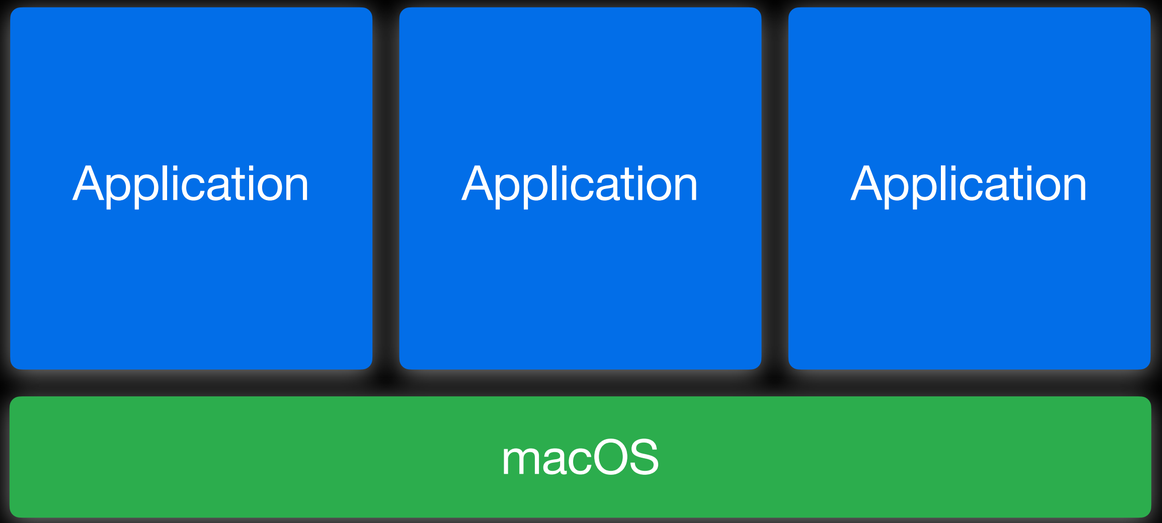
Containerization principles

Extending Container

Building on Containerization

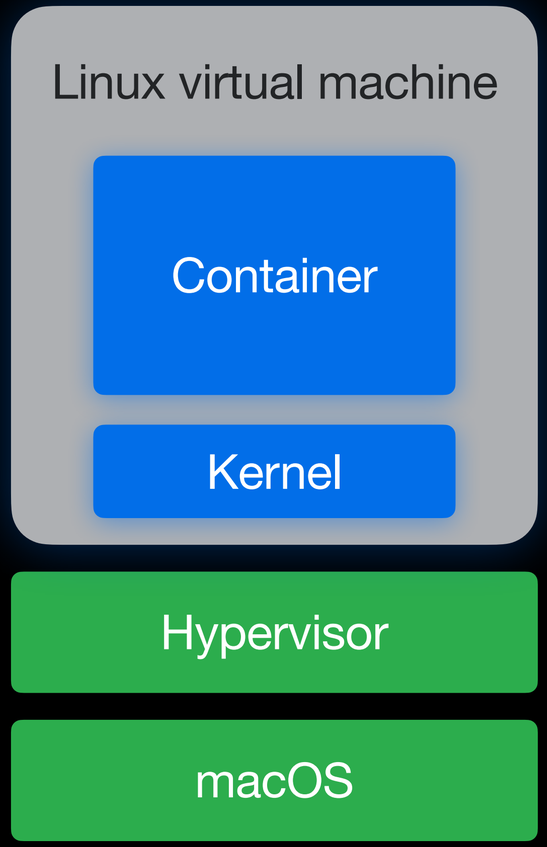
Adoption of Swift

Open Source Community



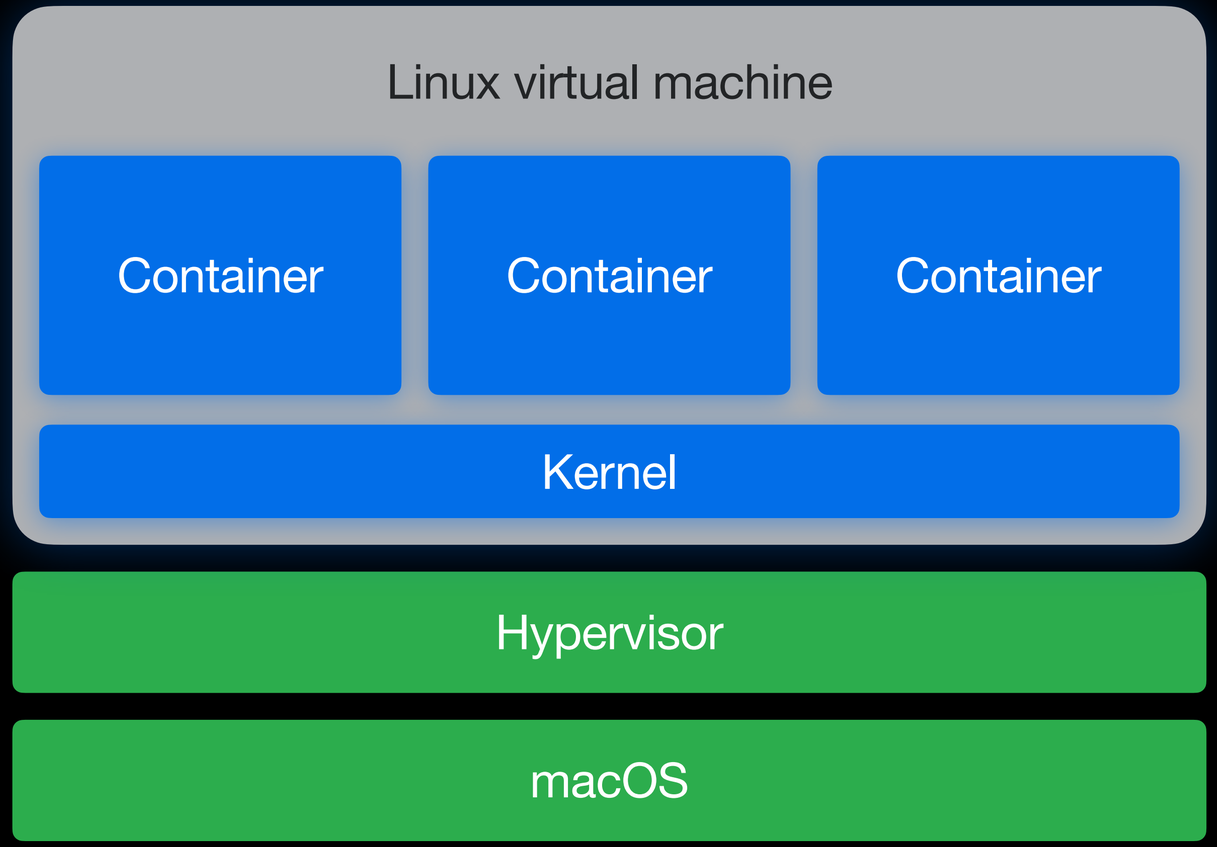
Isolation

- Memory - virtual addressing
- Process visibility - all host processes
- Storage - permissions
- Network - none



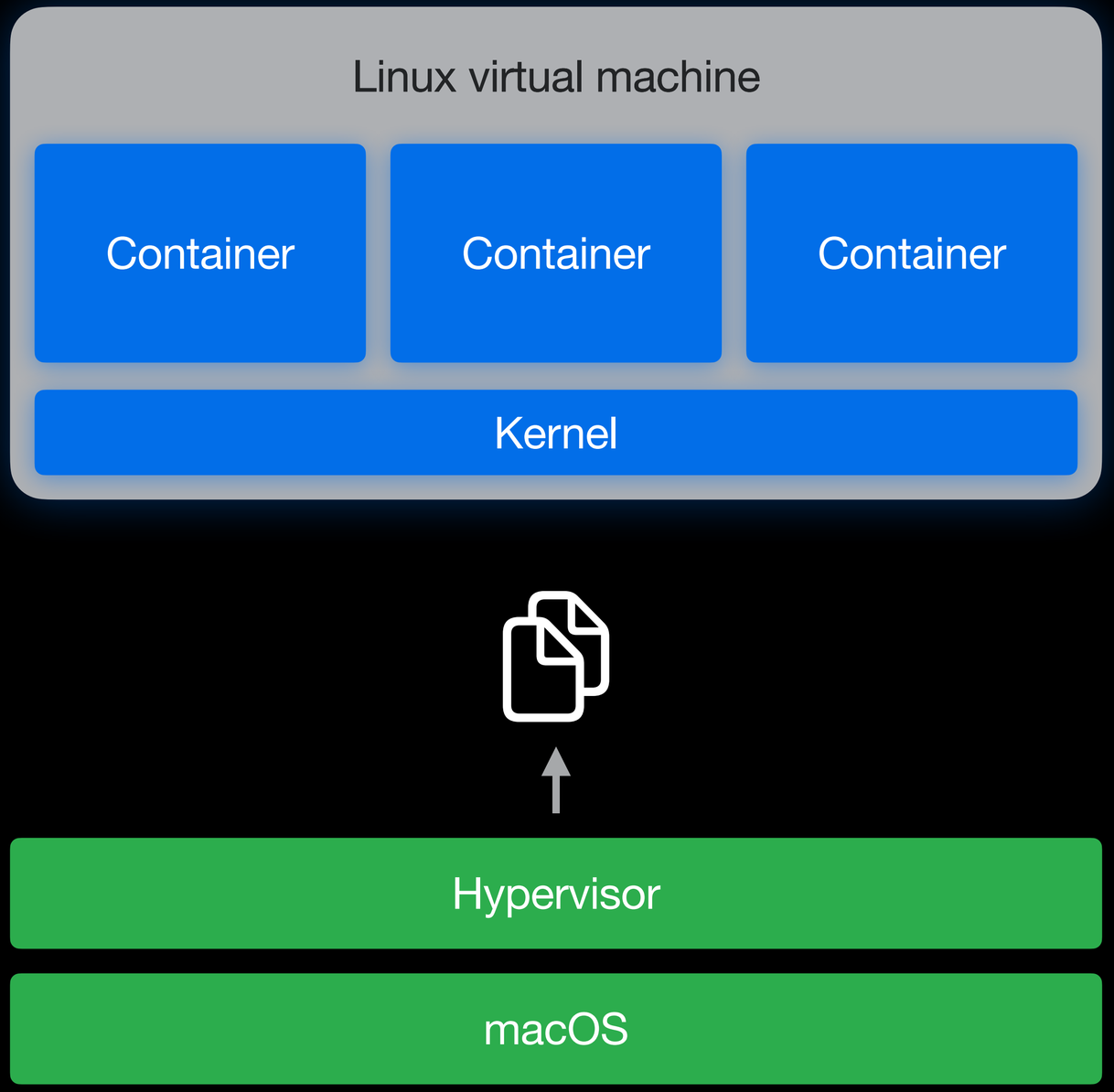
Host Isolation

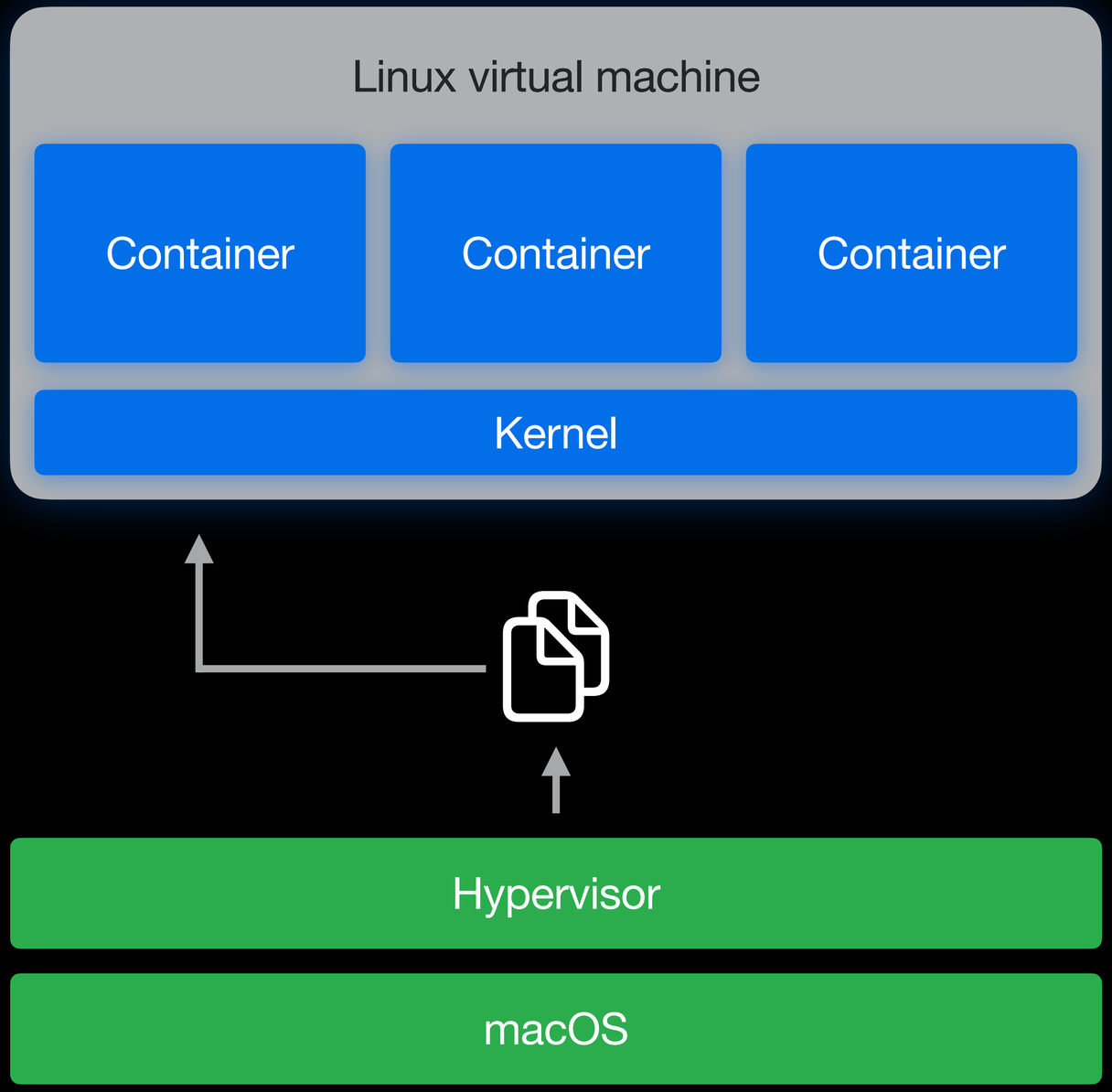
- Memory - nested paging
- Process visibility - none
- Storage - virtual devices
- Network - virtual interfaces
- Resources - VM configuration

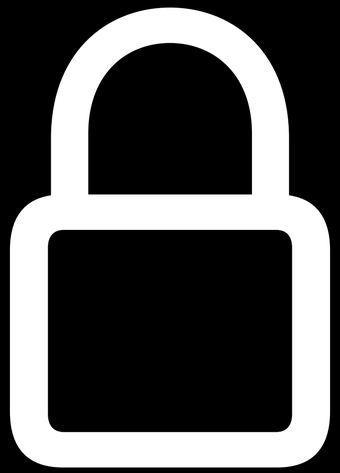


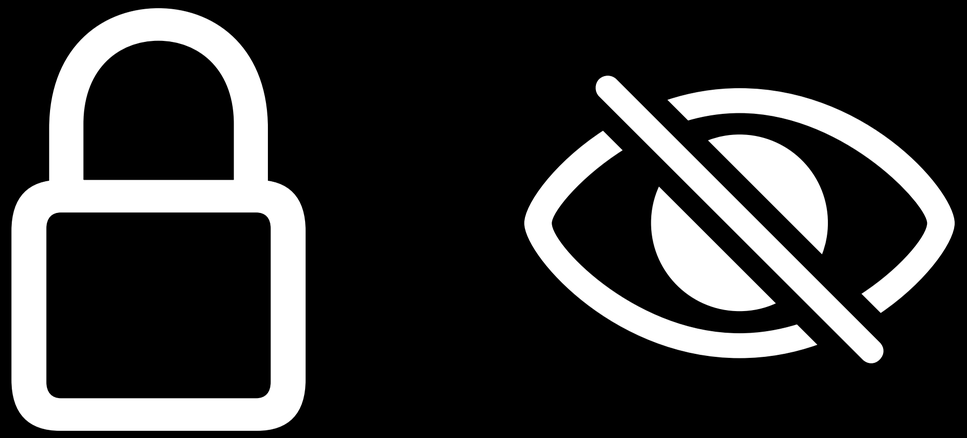
Container Isolation

- Memory - virtual addressing
- Process visibility - namespaces
- Storage - namespaces
- Network - namespaces
- Resources - cgroups

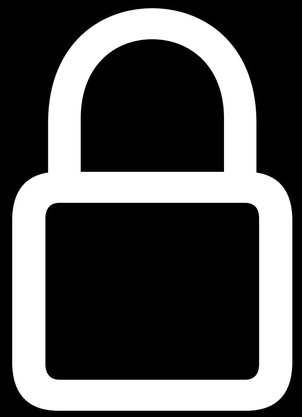

Linux virtual machine
Container
Container
Container
Kernel
Hypervisor
macOS


Linux virtual machine
Container
Container
Container
Kernel
Hypervisor
macOS



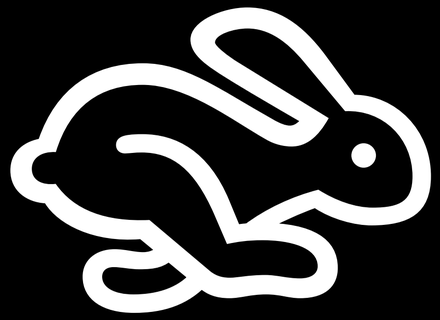


Private

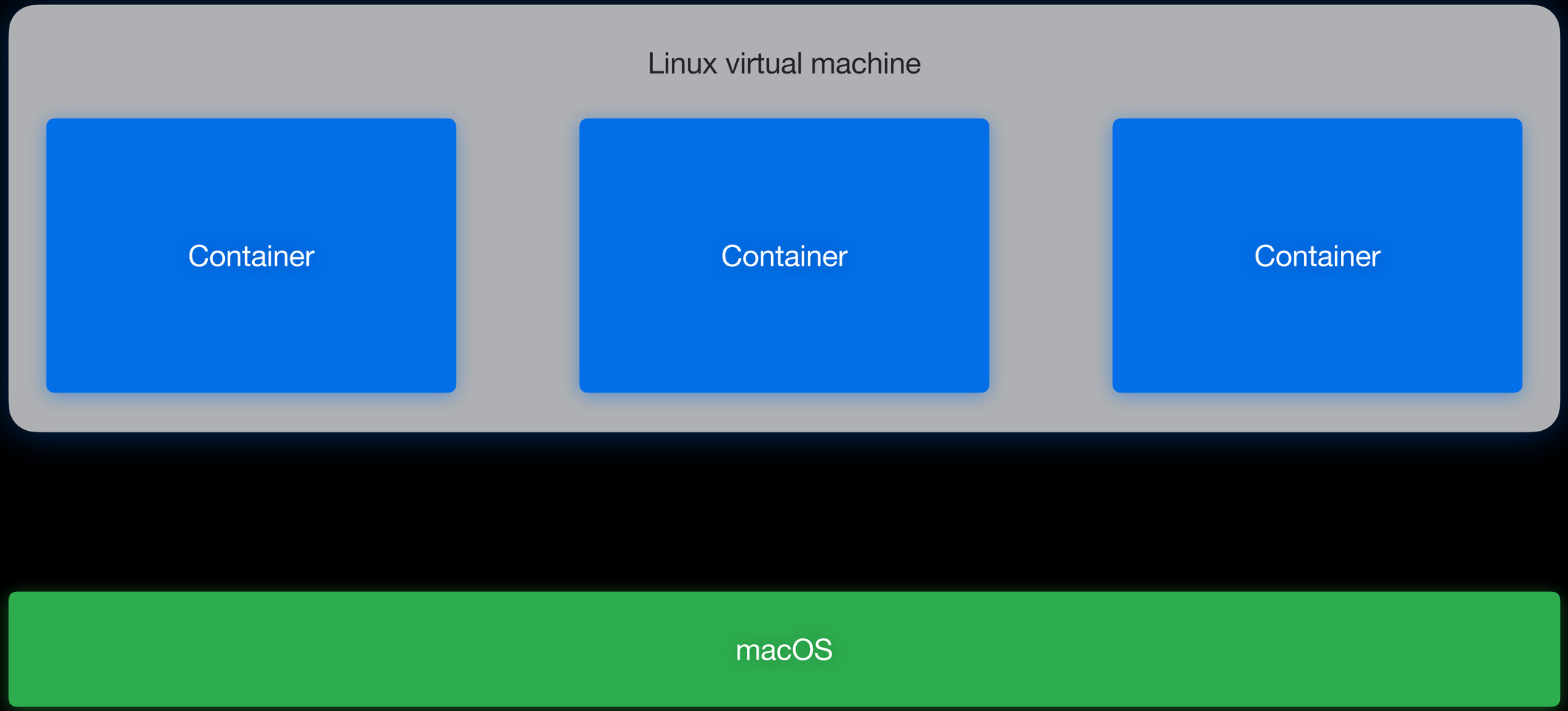


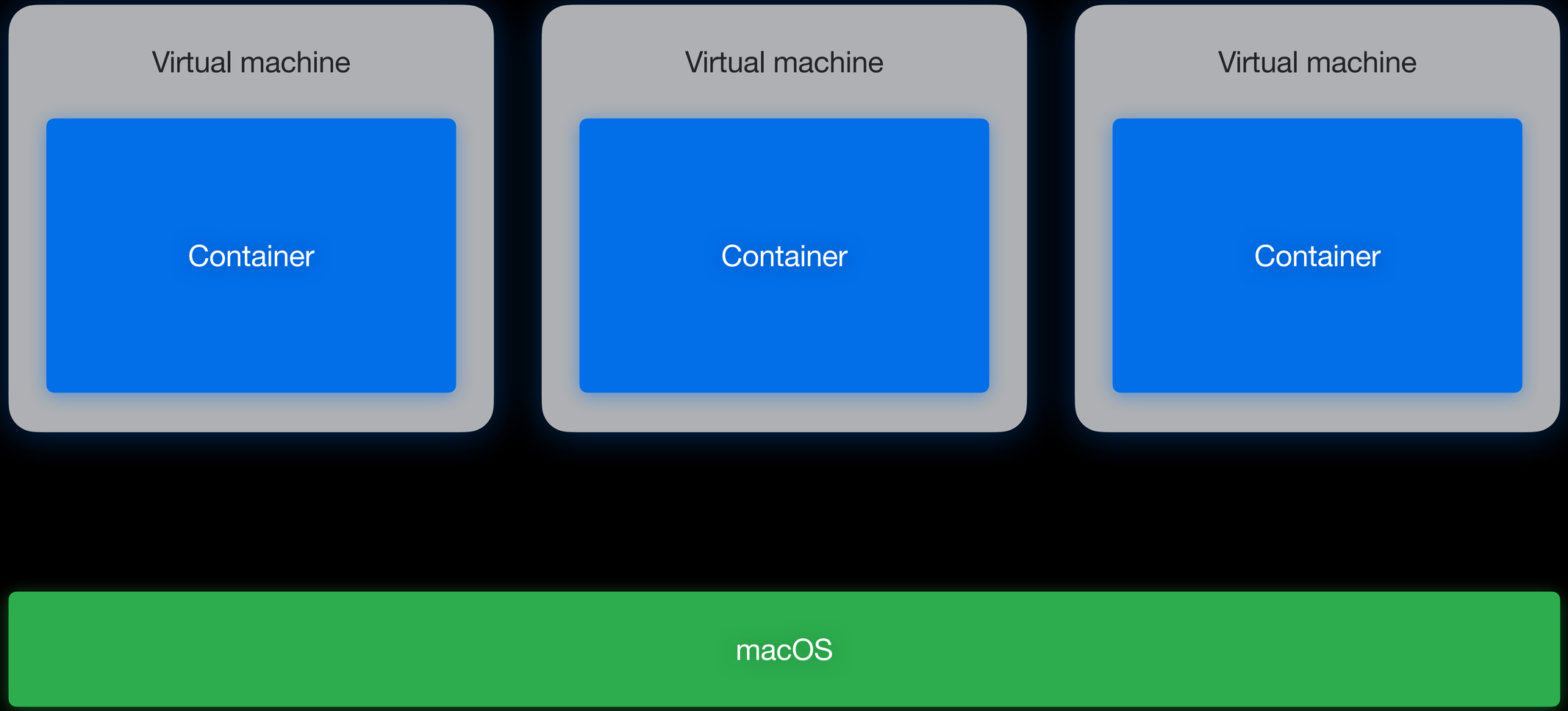
Secure

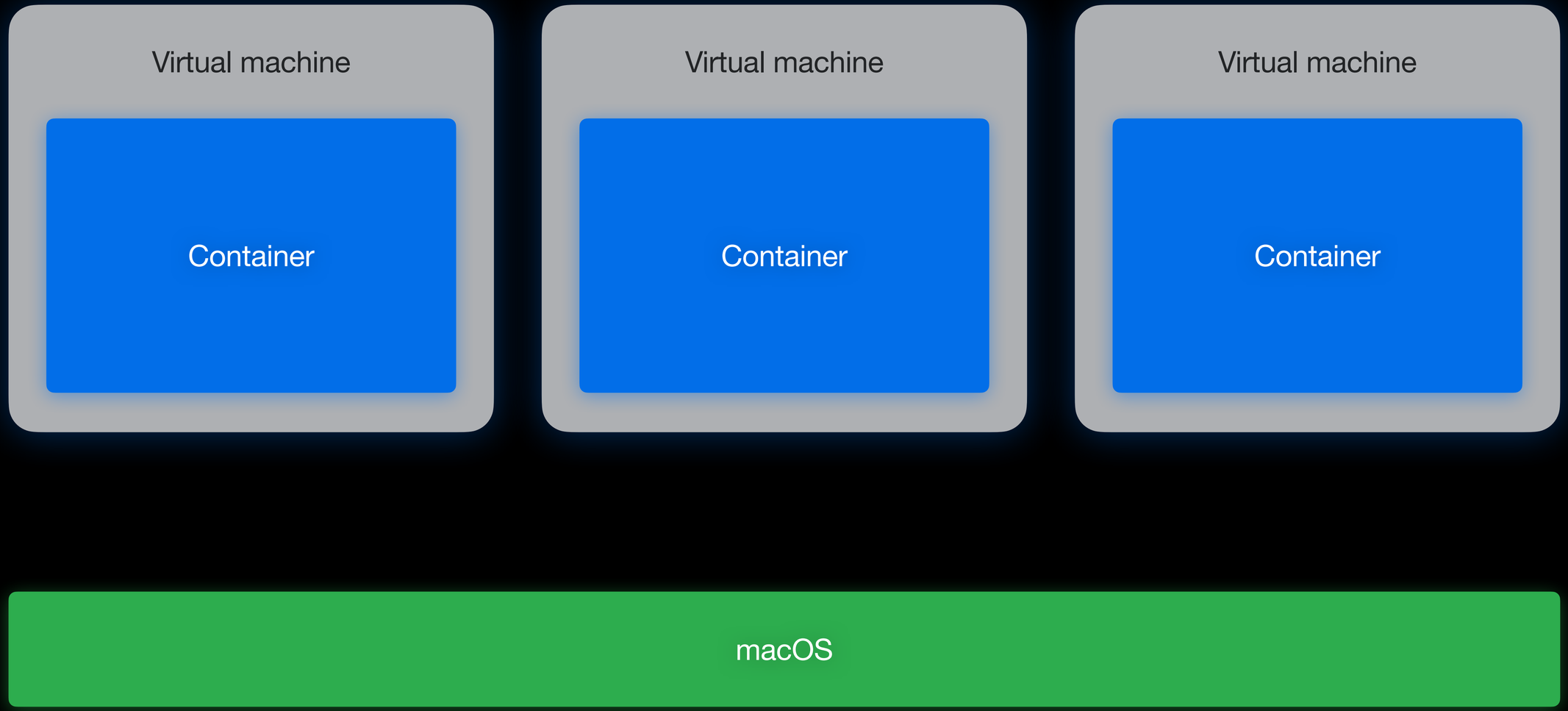
Private

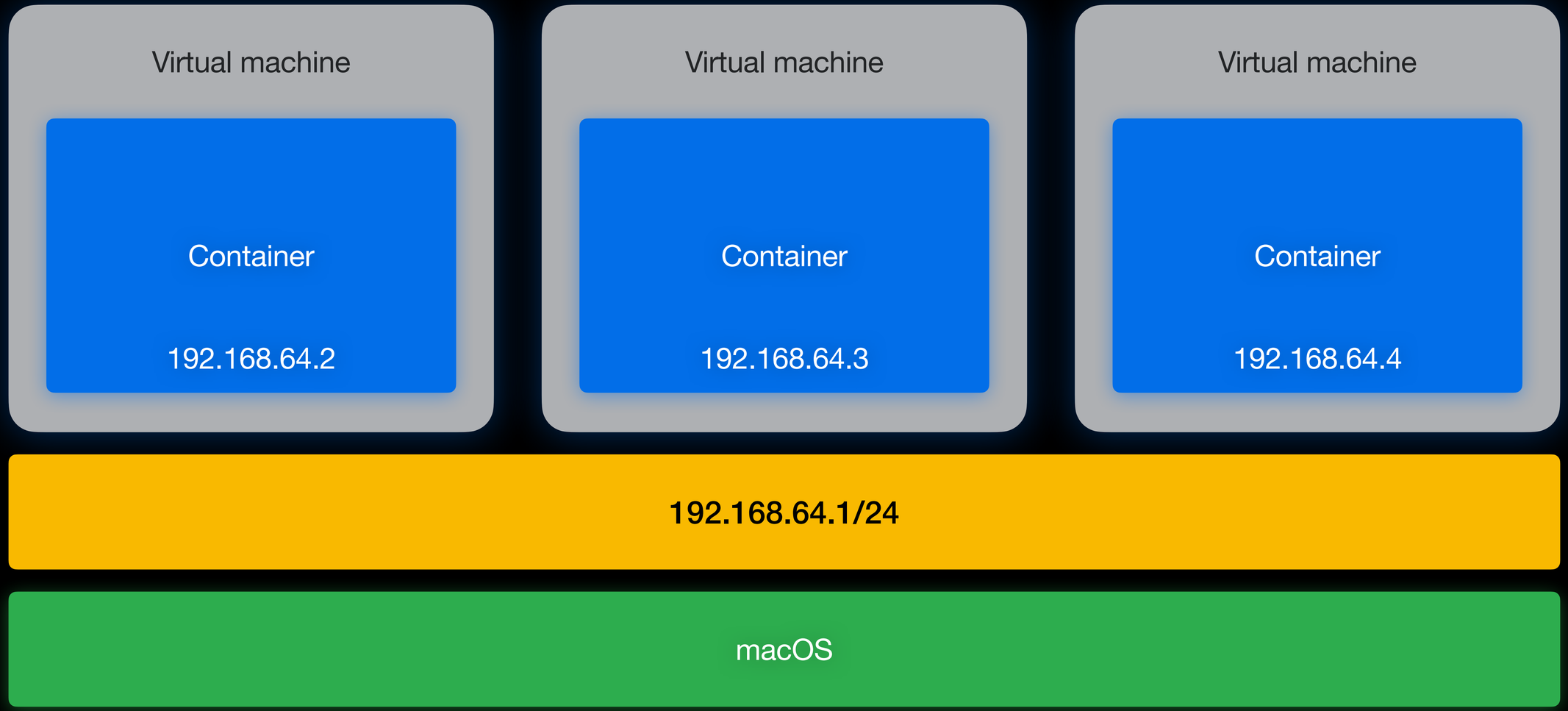


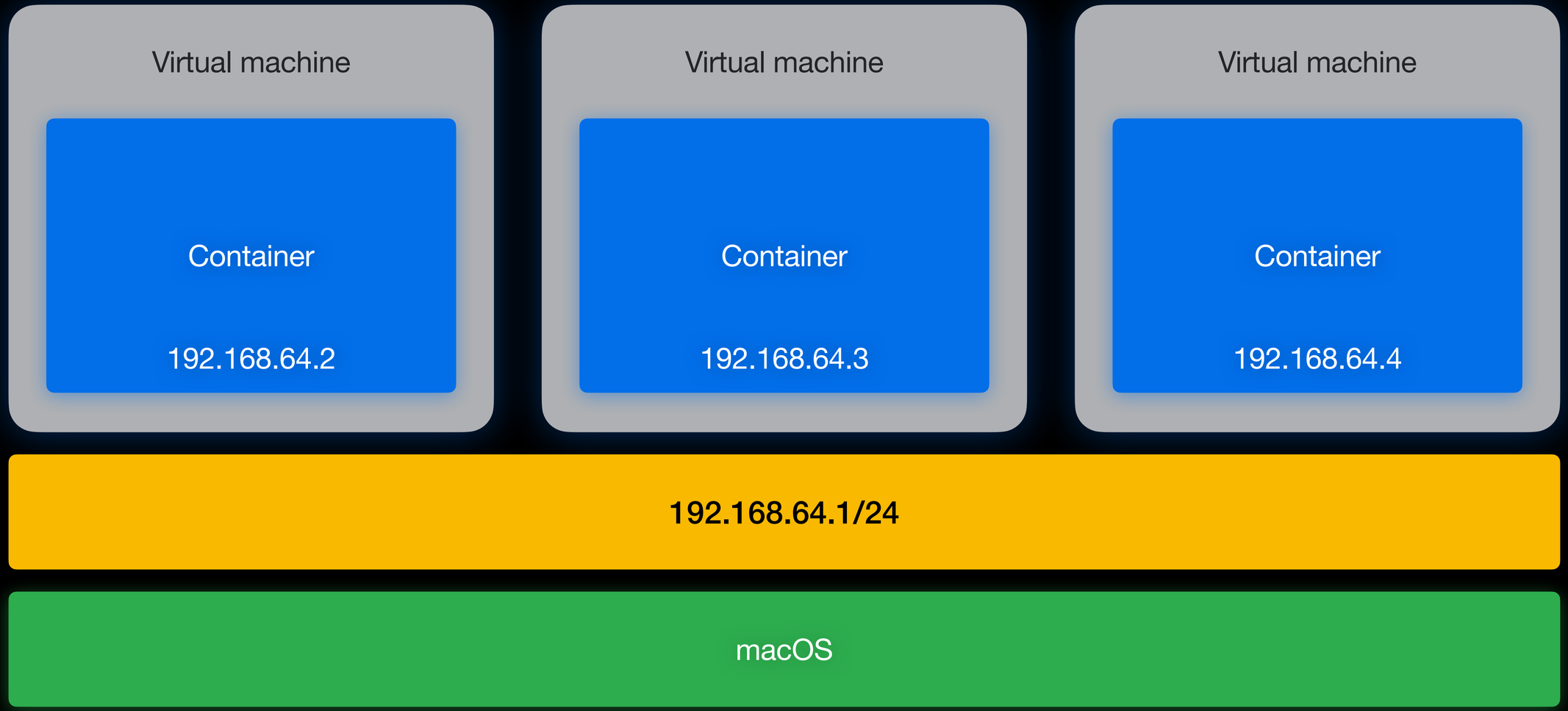
Performant

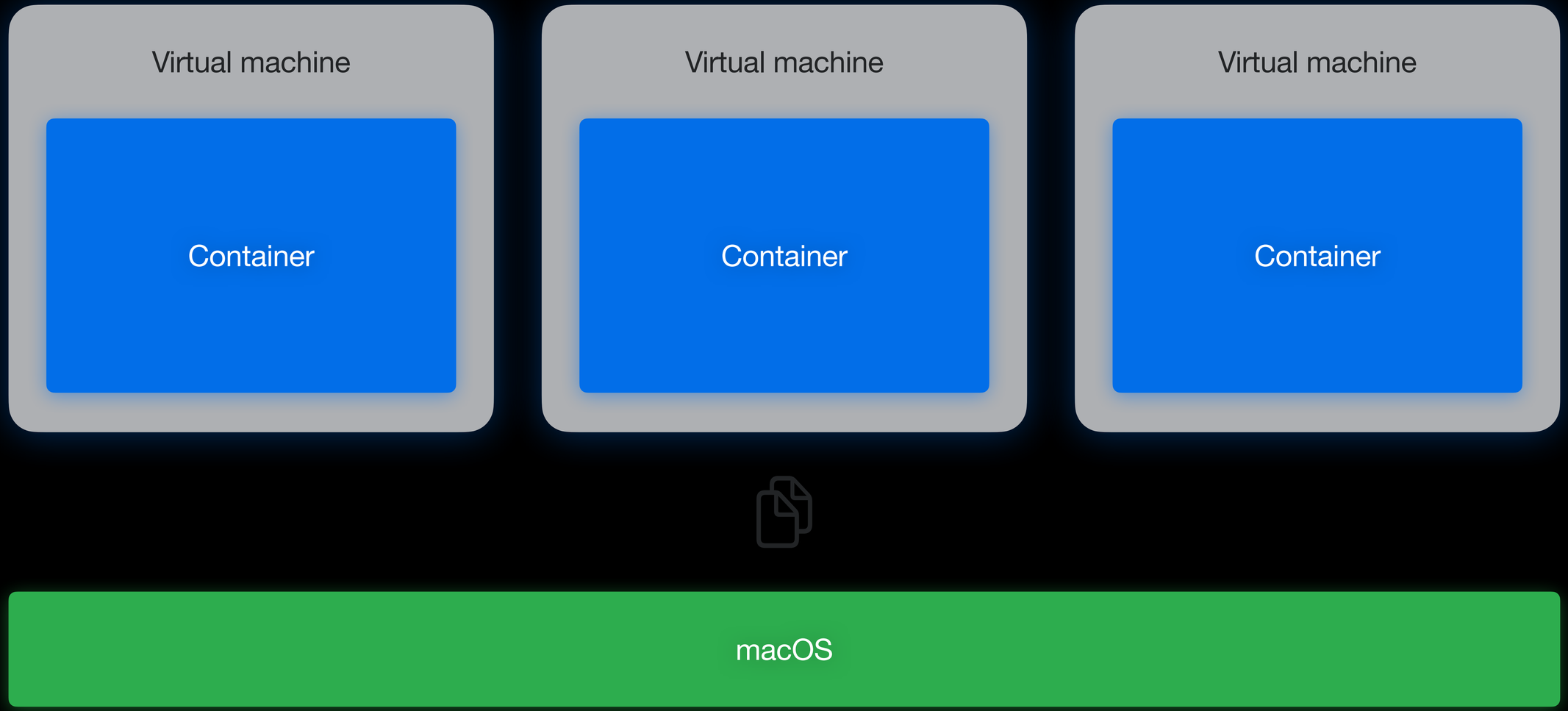

Linux virtual machine
Container
Container
Container
macOS

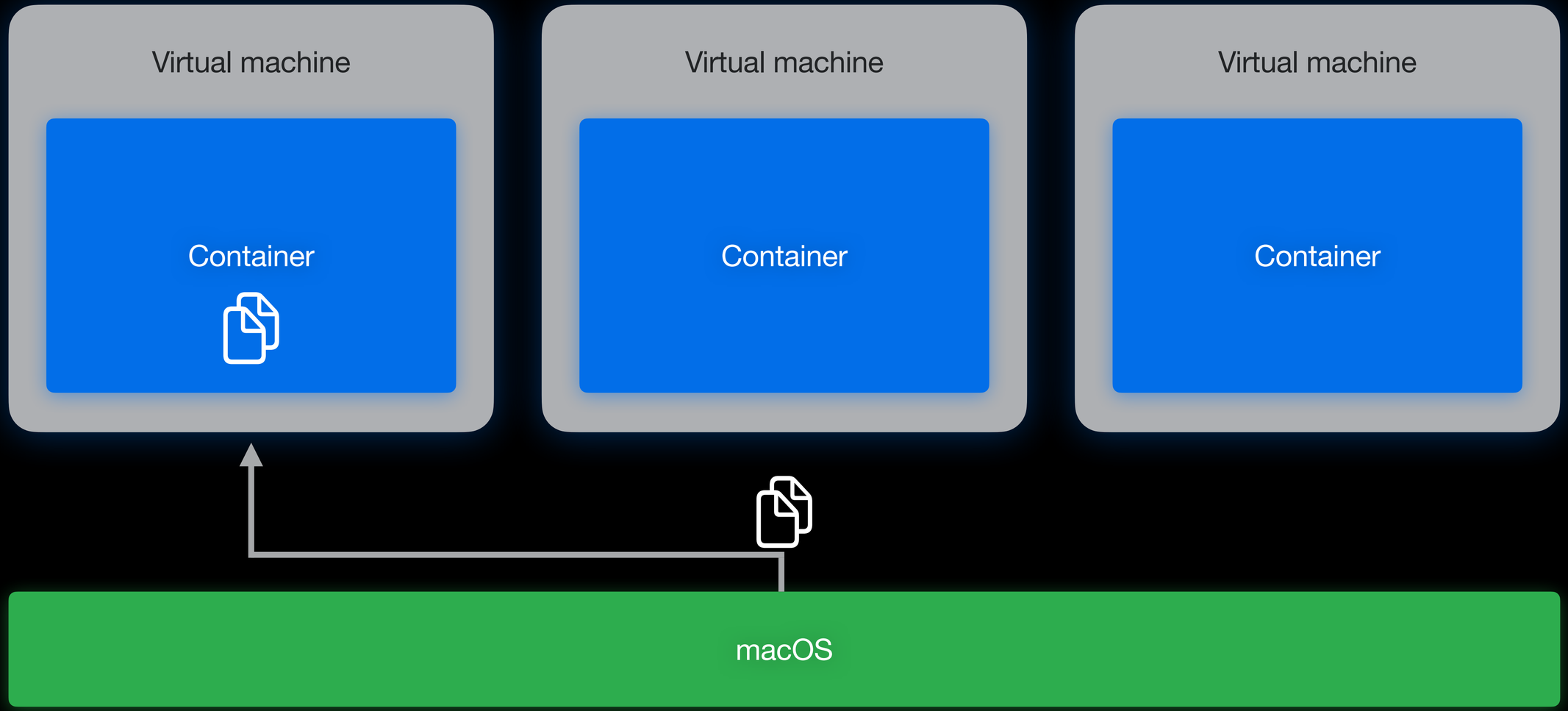

Virtual machine
Container
Virtual machine
Container
Virtual machine
Container
macOS

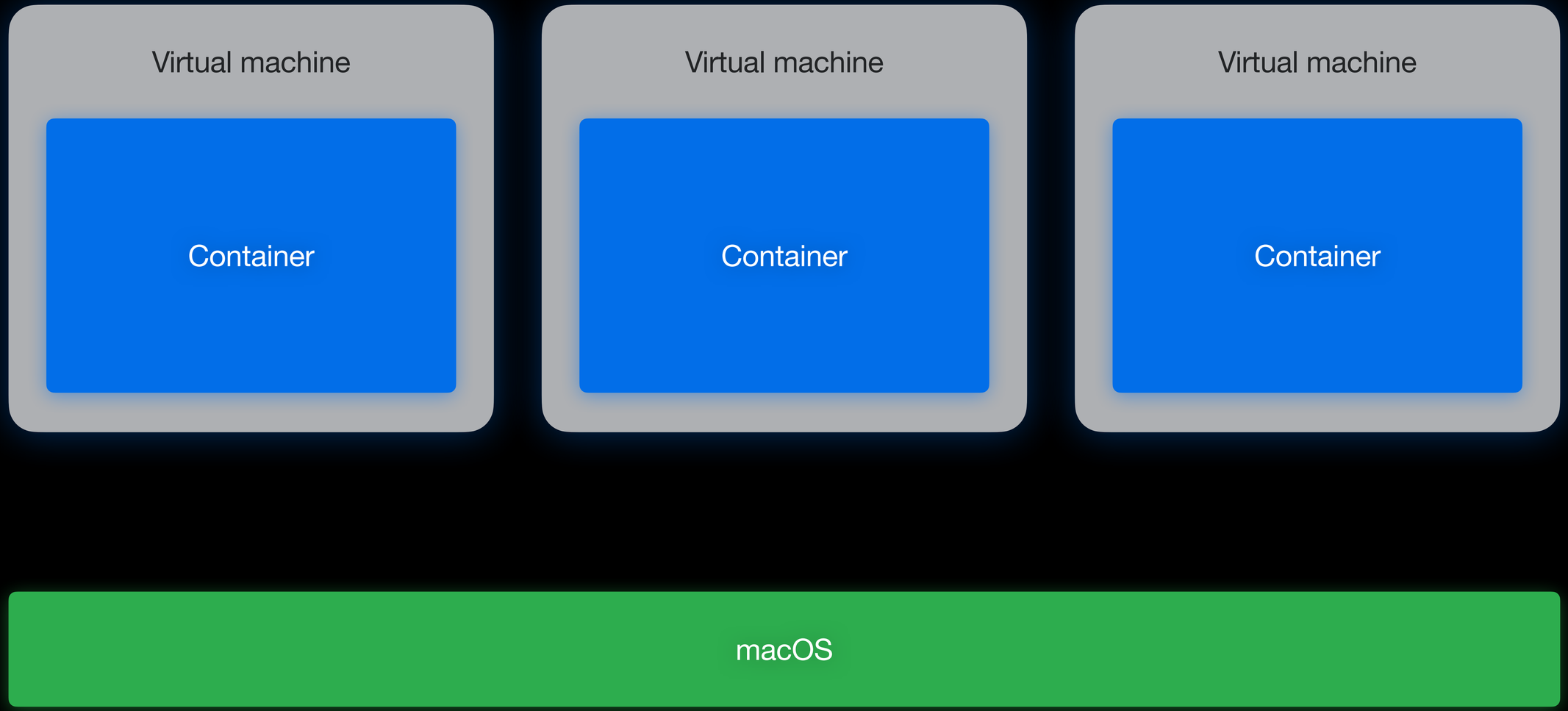

Virtual machine
Container
Virtual machine
Container
Virtual machine
Container
macOS

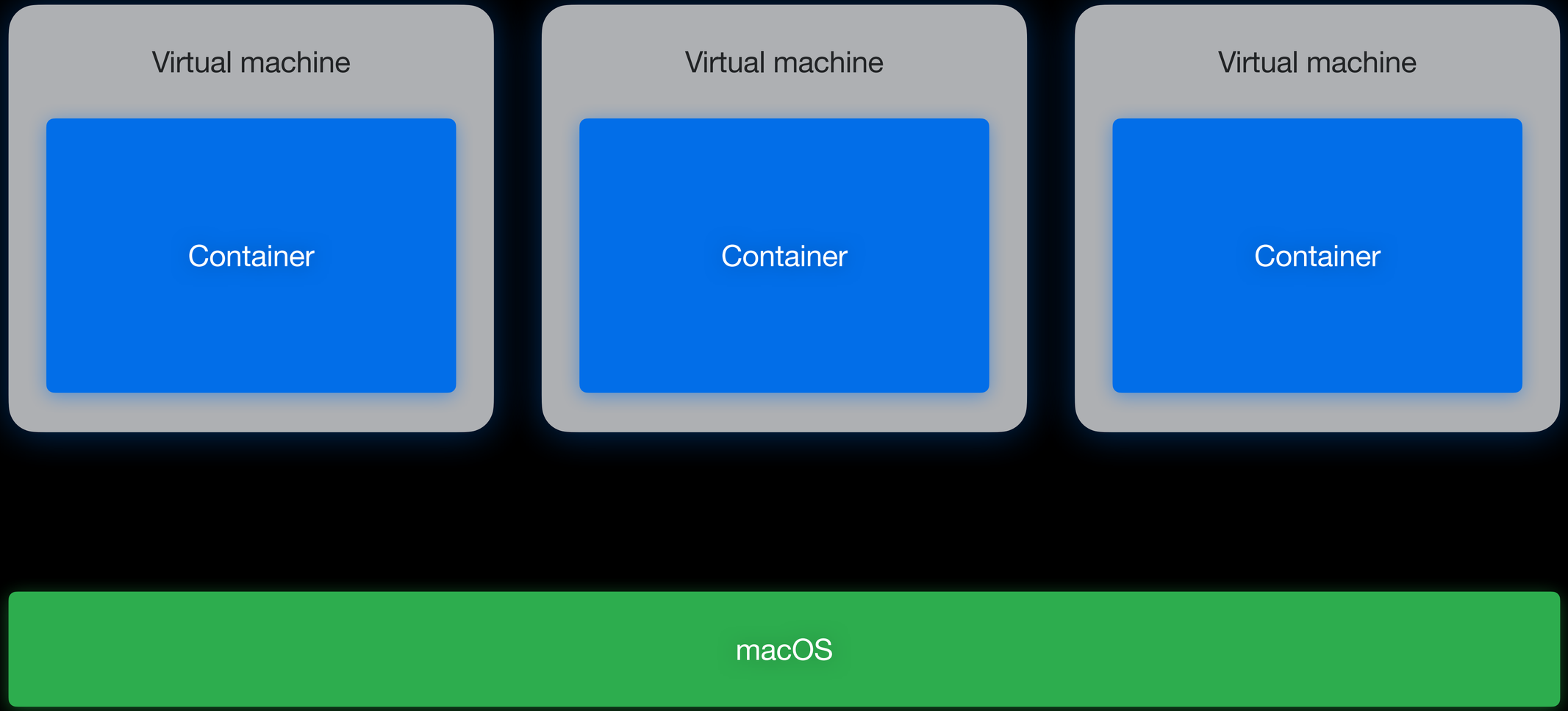

Virtual machine
Container
192.168.64.2
Virtual machine
Container
192.168.64.3
Virtual machine
Container
192.168.64.4
192.168.64.1/24
macOS


Virtual machine
Container
192.168.64.2
Virtual machine
Container
192.168.64.3
Virtual machine
Container
192.168.64.4
192.168.64.1/24
macOS

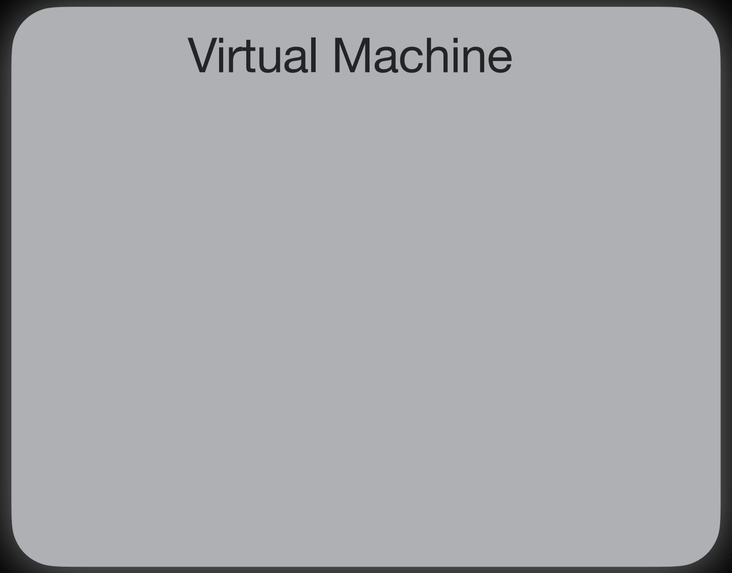

Virtual machine
Container
Virtual machine
Container
Virtual machine
Container
macOS


Virtual machine
Container
Virtual machine
Container
Virtual machine
Container
macOS


Virtual machine
Container
Virtual machine
Container
Virtual machine
Container
macOS

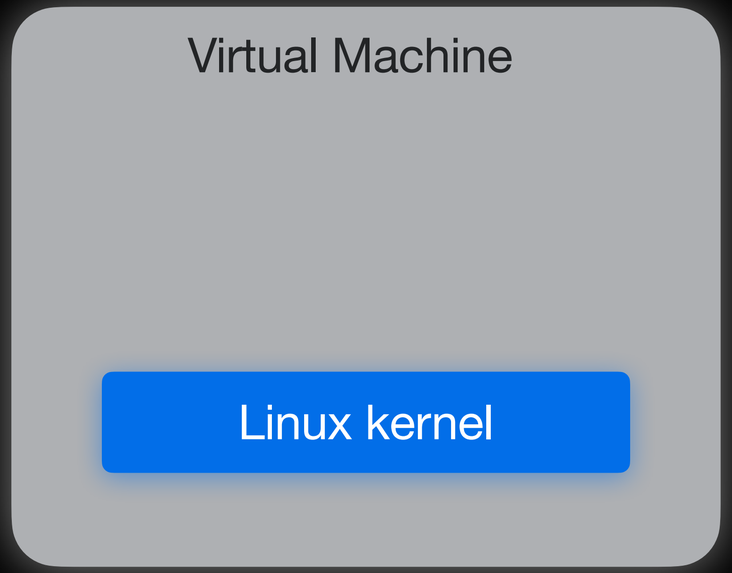

Virtual machine
Container
Virtual machine
Container
Virtual machine
Container
macOS

Lightweight Virtual Machine



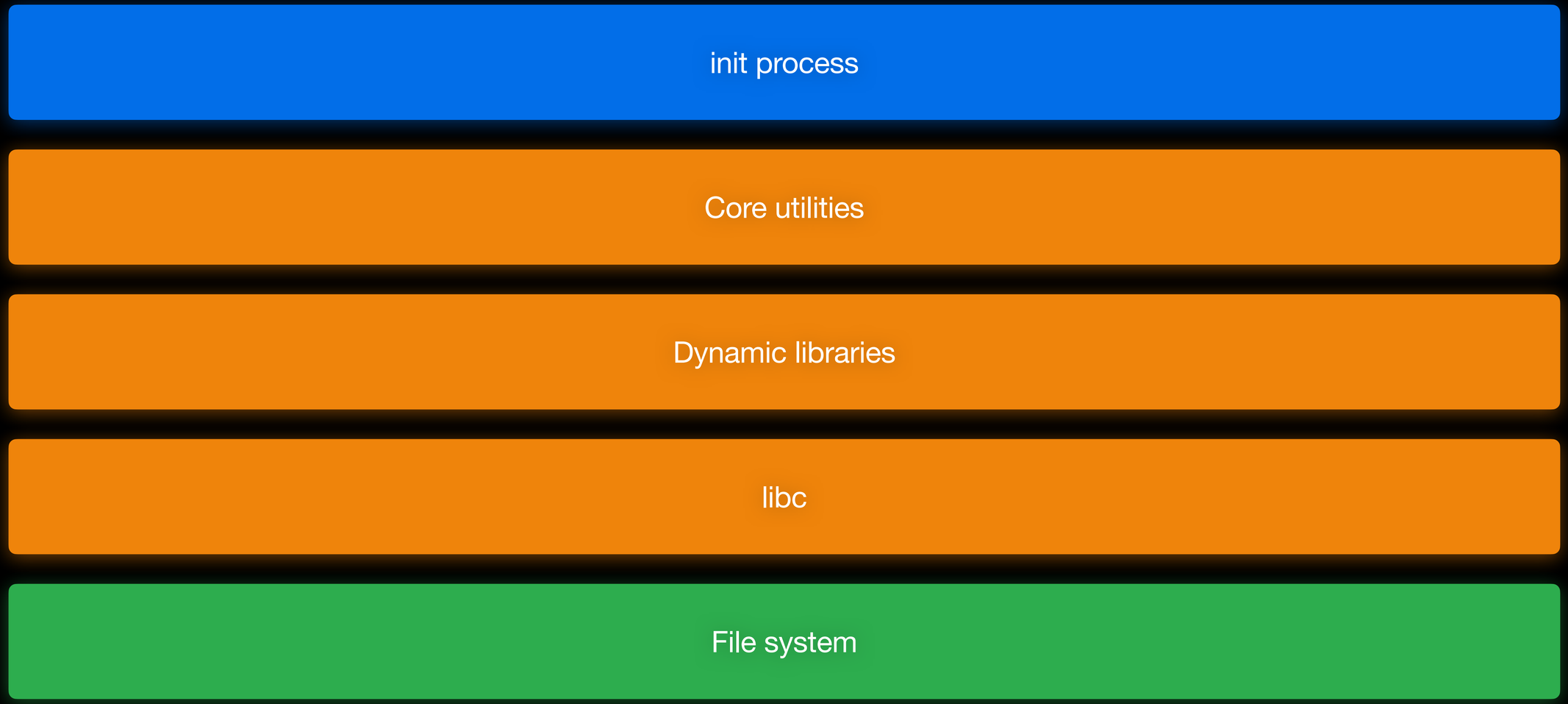
- ✓ minimal devices
- ✓ right sized memory
- ✓ right sized CPU

Lightweight Virtual Machine

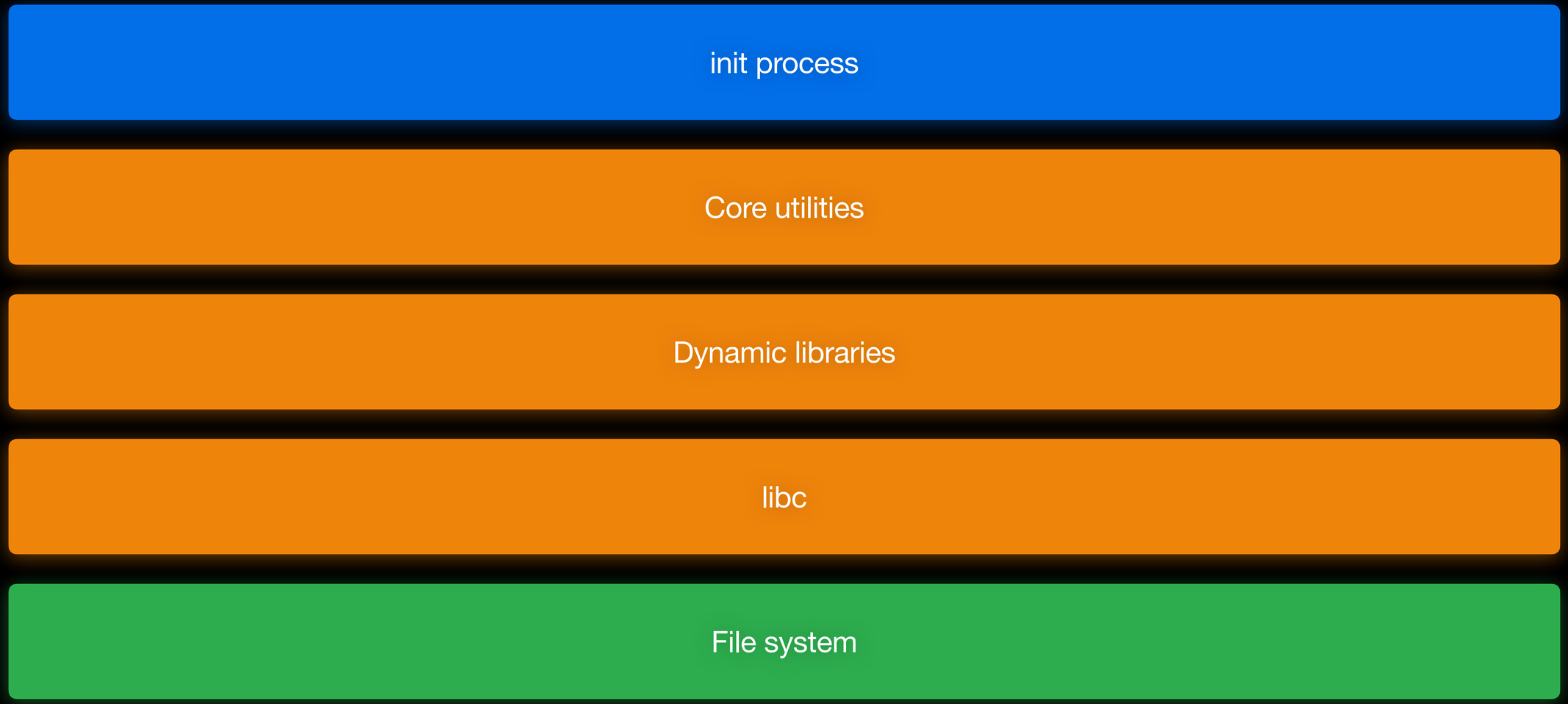


- ✓ direct boot kernel
- ✓ minimal kernel config

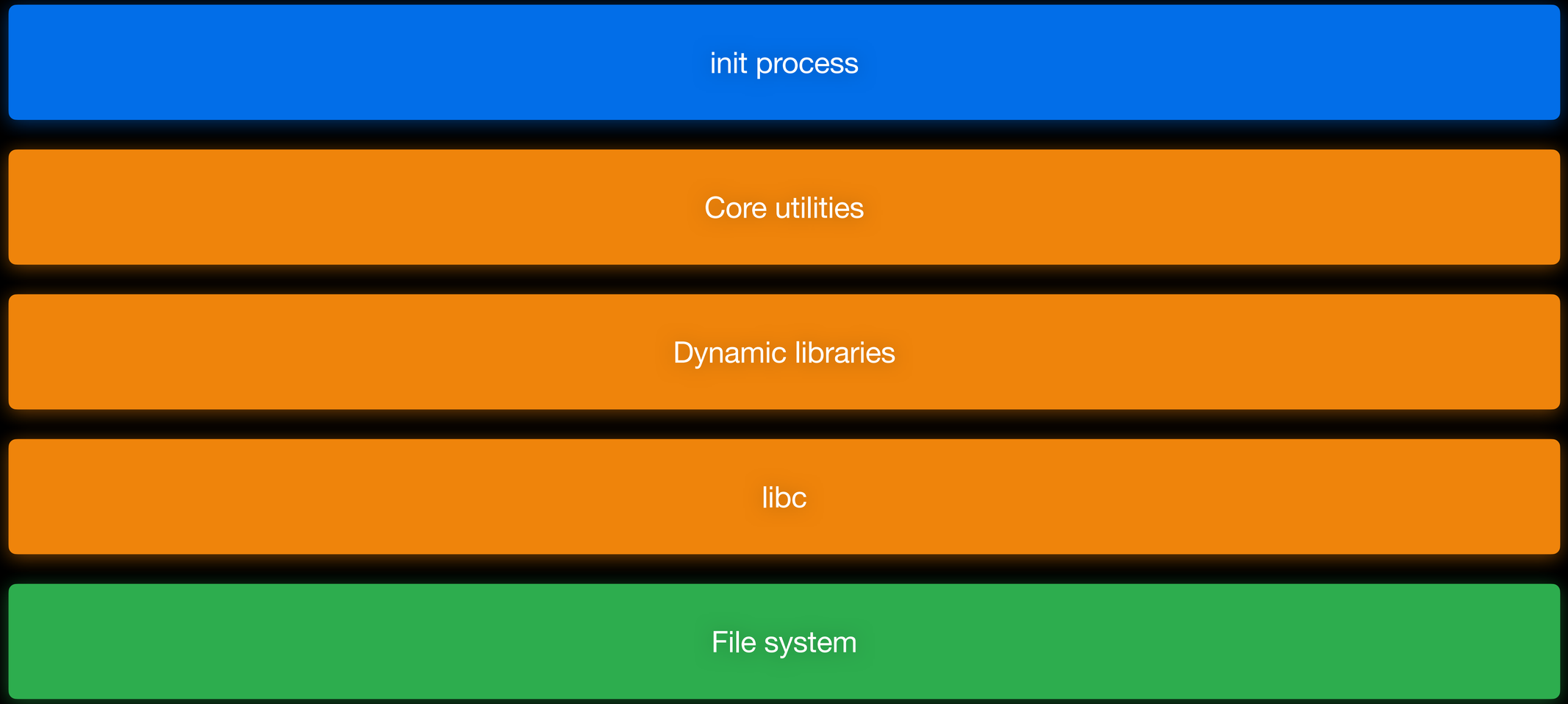
Typical initial filesystem



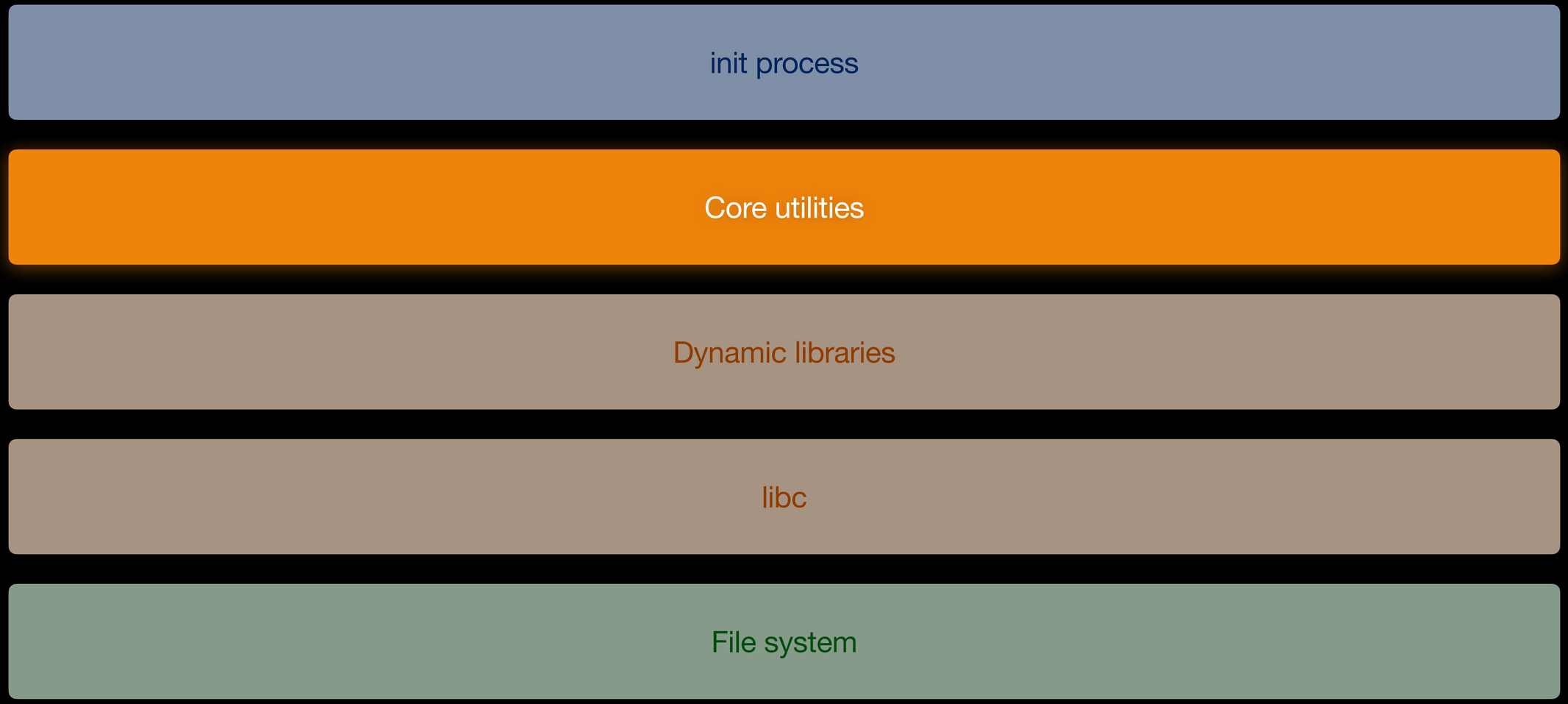
Typical initial filesystem



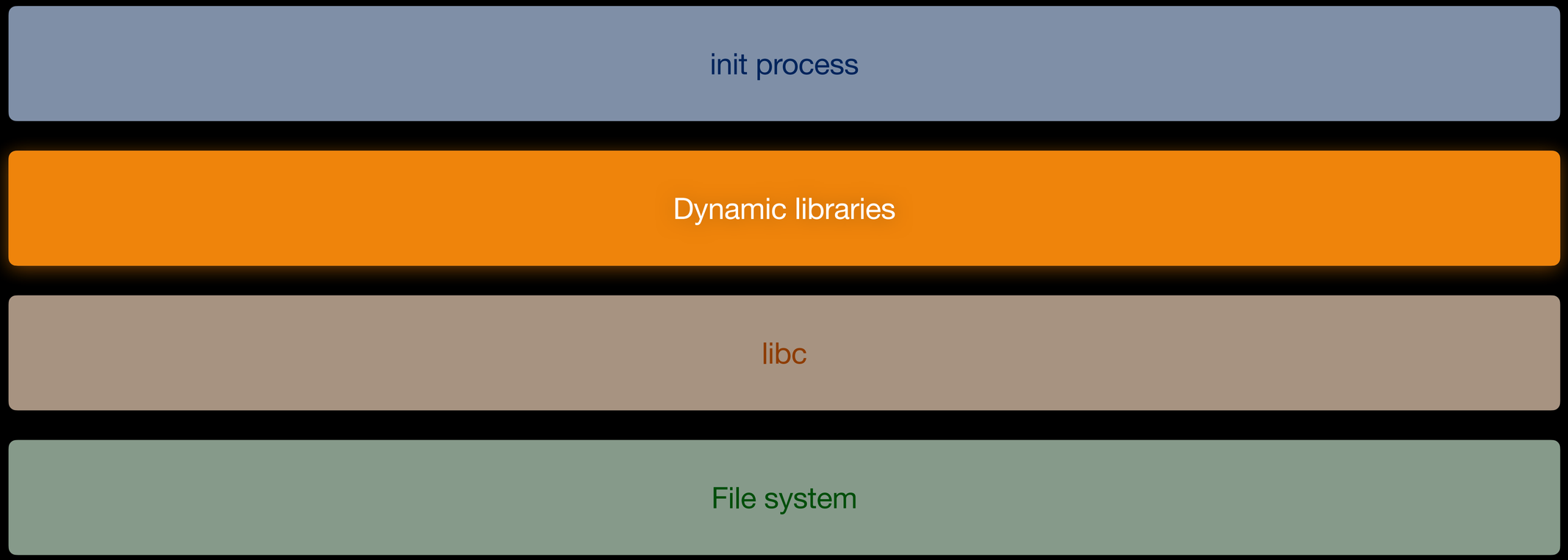
Typical initial filesystem



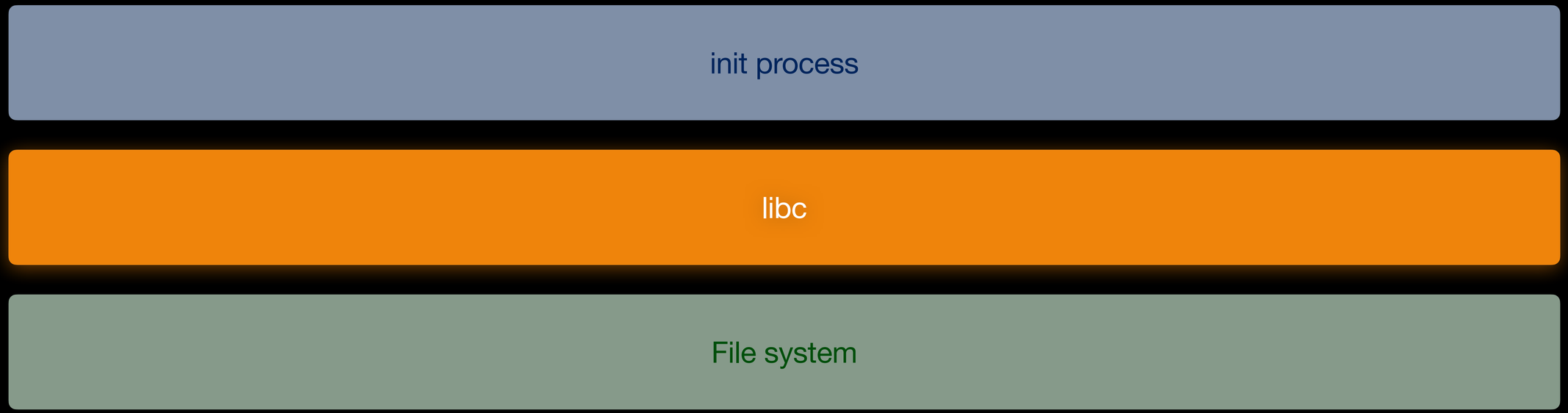
Typical initial filesystem



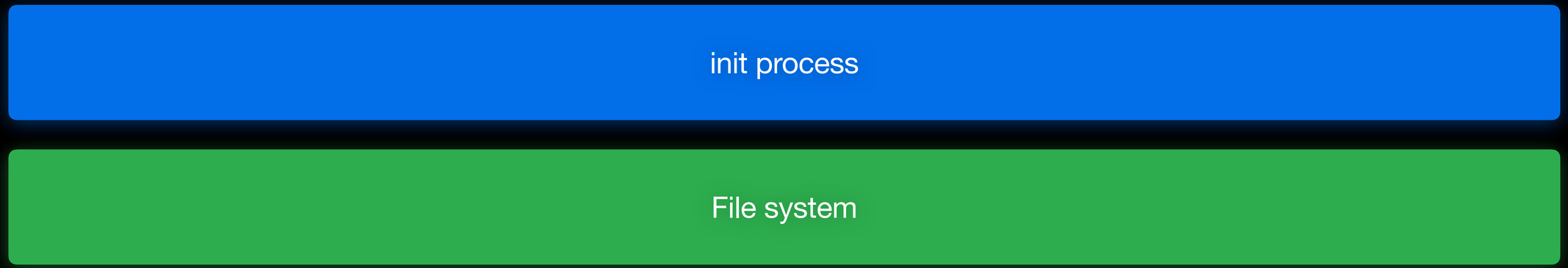
Typical initial filesystem



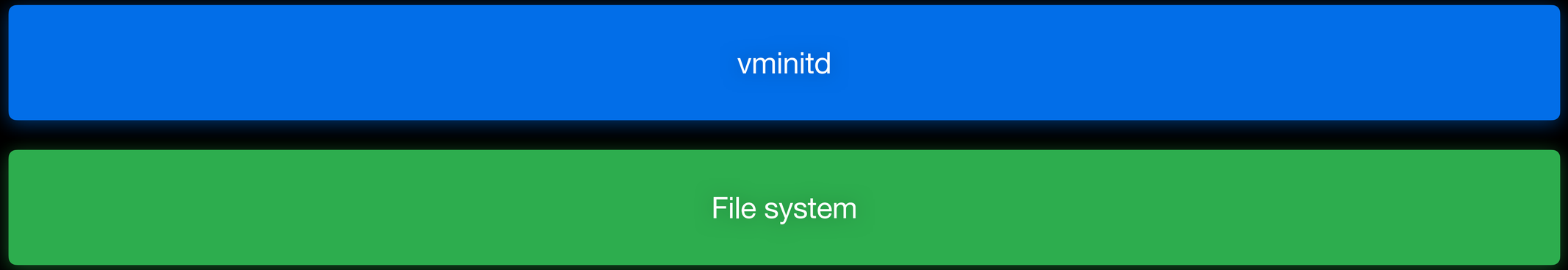
Typical initial filesystem



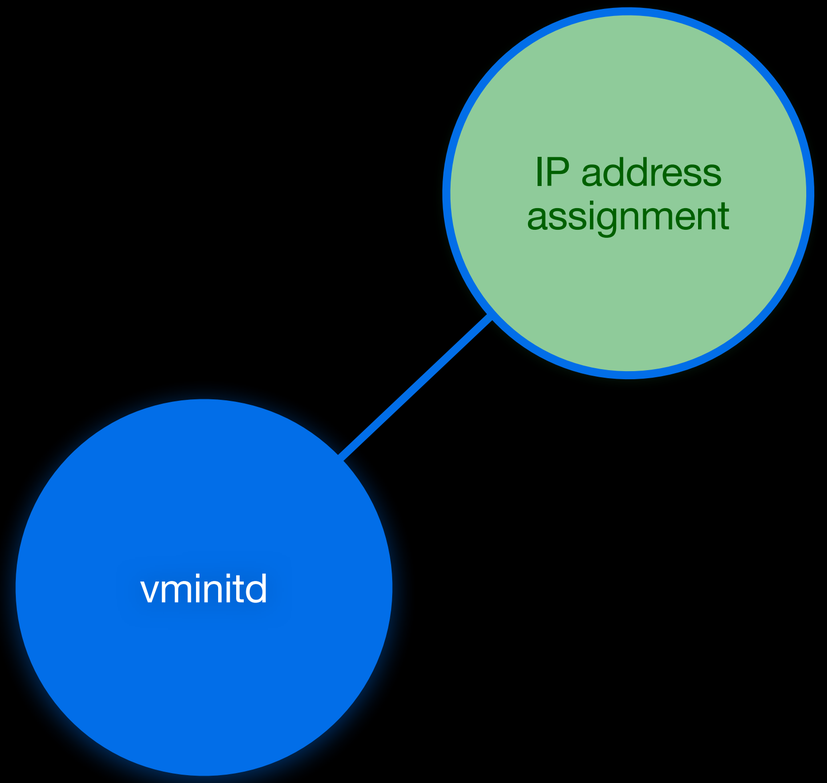
Containerization init filesystem

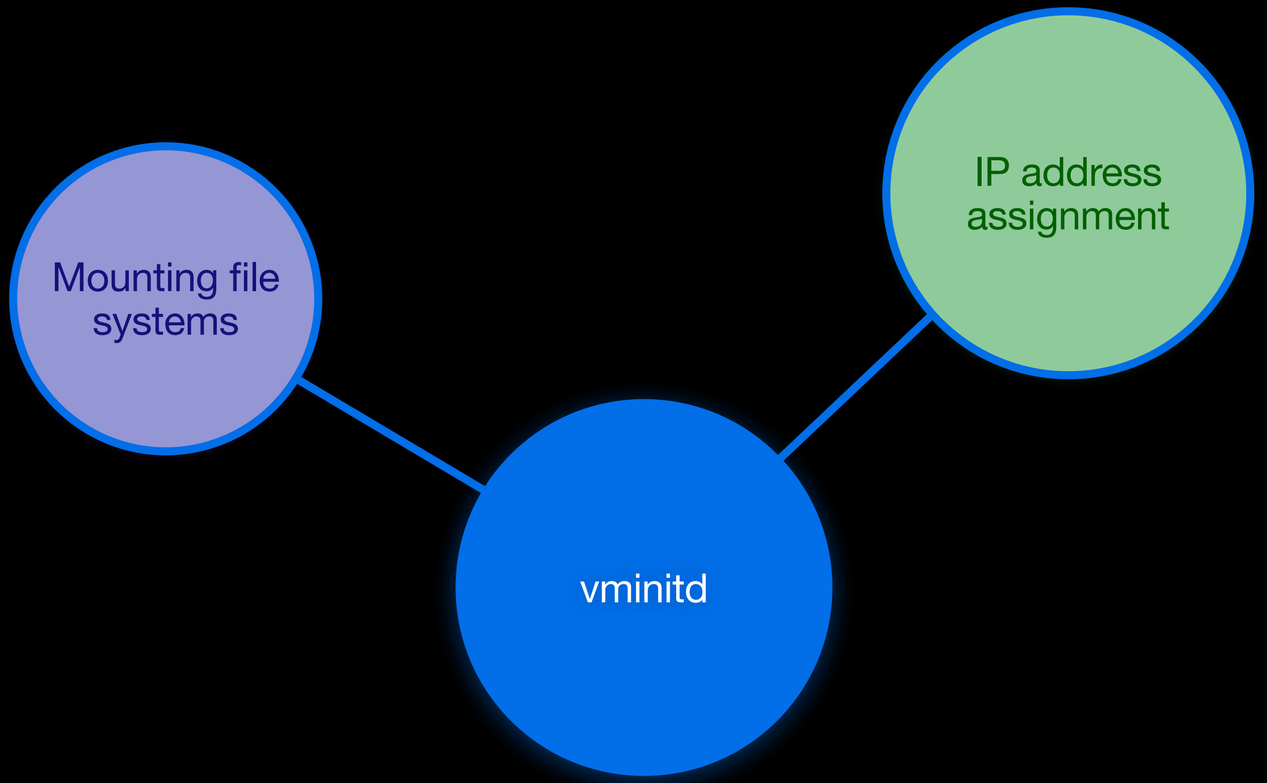


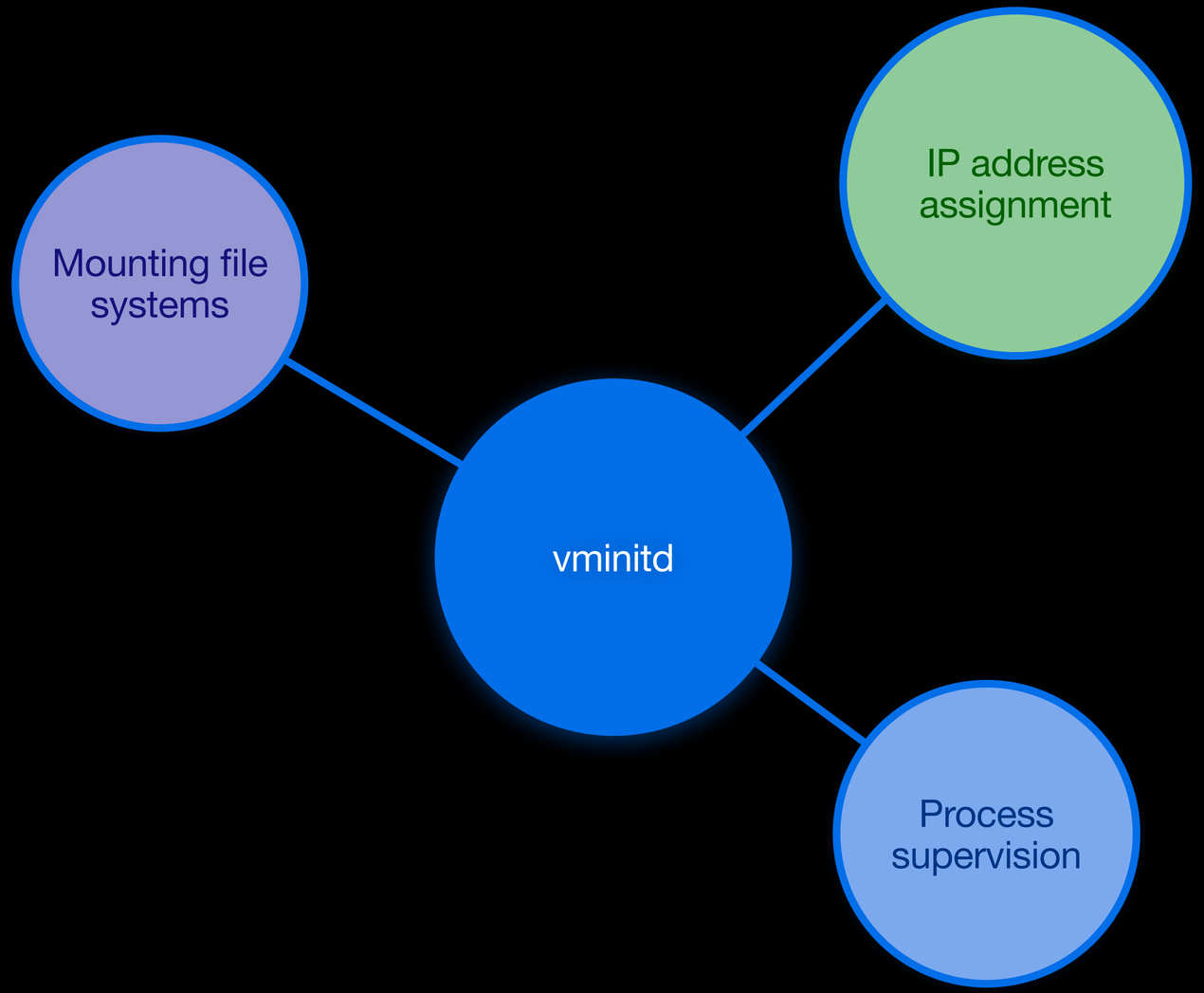
Containerization init filesystem

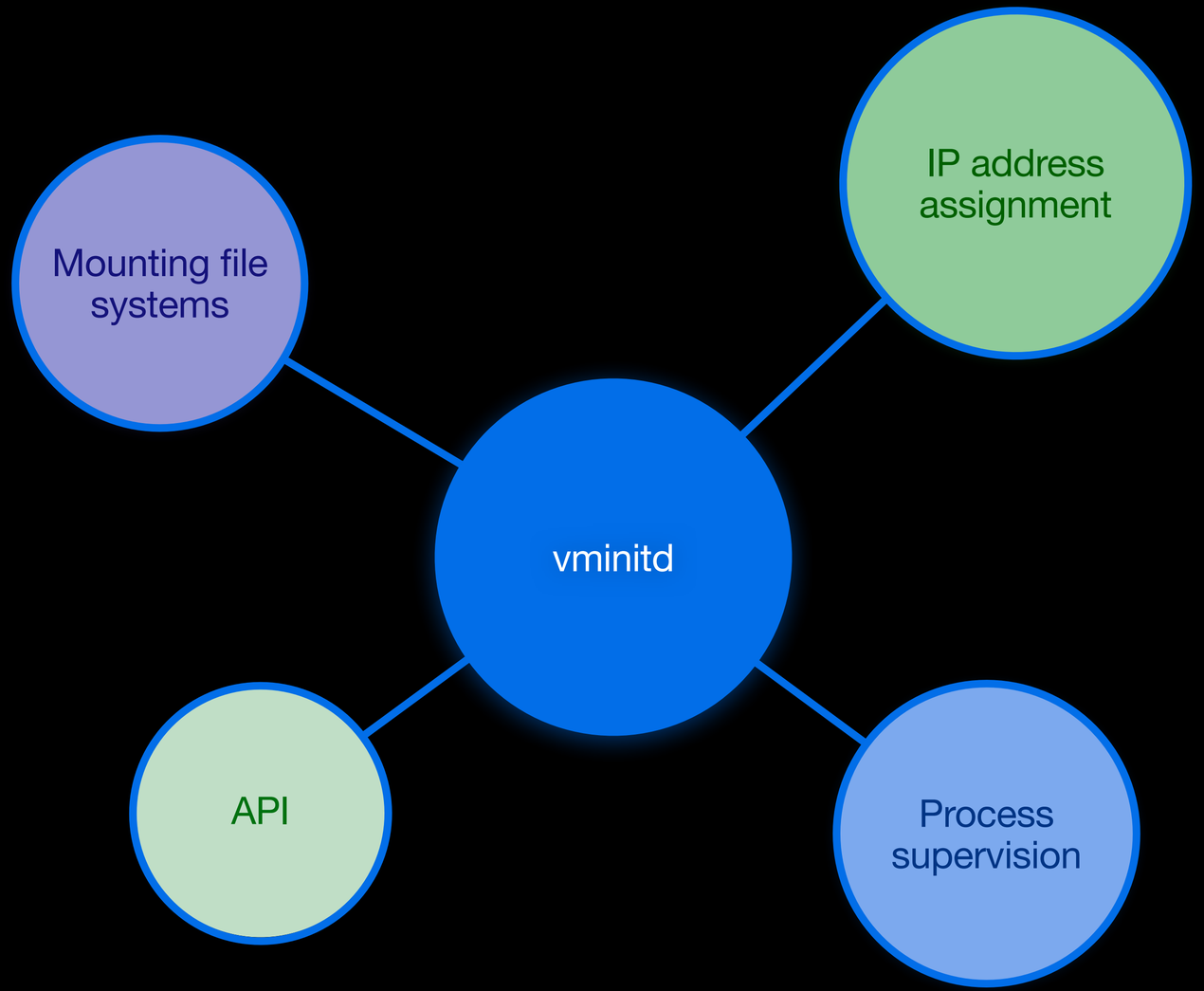


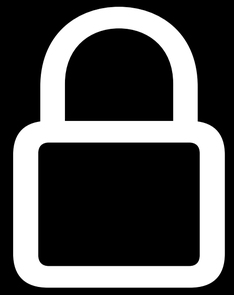
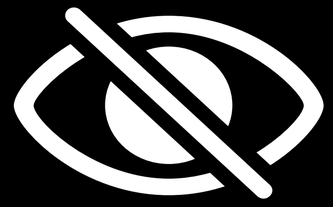
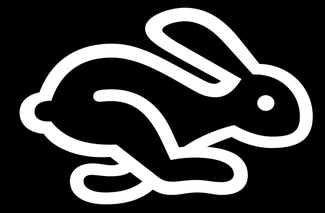

vminitd


IP address
assignment
vminitd


Mounting file systems
IP address assignment
vminitd


Mounting file systems
IP address assignment
vminitd
Process supervision


Mounting file systems
IP address assignment
vminitd
API
Process supervision

Minimal kernel	✓		✓
Direct boot			✓
Static init process purpose built for containers	✓		✓
Each workload in its own virtual machine	✓	✓	✓

What are containers?

Containerization principles

Extending Container

Building on Containerization

Adoption of Swift

Open Source Community

Container plugin framework

- Enable developers to extend functionality without modifying core code.
- Types of plugins:
  - CLI plugins: extend command line with new subcommands
  - Daemon plugins: background services

- Plugins used for key system components today:

```
container-runtime-linux | container-core-images | container-network-vmnet
```

Developing your own plugin

Automatic plugin discovery

Simple structure:

```
plugin-name/
|- bin/plugin-name
|- config.json

cat config.json
{
  “abstract”: “Your CLI plugin description”
  “version”: “1.0”,
  “author”: “your name”
}
```

What are containers?

Containerization principles

Extending Container

Building on Containerization

Adoption of Swift

Open Source Community

Containerization: building blocks for containers

- Key packages:

```
|- Containerization
|- ContainerizationOS
|- ContainerizationNetlink
|- ContainerizationOCI
|- ContainerizationEXT4
|- ContainerizationIO
```

Protocol-based
Layered Architecture

Containerization: layered architecture

High-Level

Low-Level

Container Orchestration
ContainerManager | LinuxContainer | LinuxPod | ImageStore

Process + VM Control
LinuxProcess | VirtualMachineInstance | Vminitd | SocketRelay

Primitives
Configuration: Mount, Interface, DNS, RLimit
Storage: Image, Unpacker, EXT4
OS Utilities: Command, Terminal, Networking

Demonstration

What are containers?

Containerization principles

Extending Container

Building on Containerization

Adoption of Swift

Open Source Community

Using Swift

- C Interoperability

Using Swift

- C Interoperability
- Enums/tagged unions and optionals

Using Swift

- C Interoperability
- Enums/tagged unions and optionals
- Memory safety

Using Swift

- C Interoperability
- Enums/tagged unions and optionals
- Memory safety
- Concurrency checking

Using Swift

- C Interoperability
- Enums/tagged unions and optionals
- Memory safety
- Concurrency checking
- Static Linux SDK

Using Swift

- C Interoperability
- Enums/tagged unions and optionals
- Memory safety
- Concurrency checking
- Static Linux SDK
- Native integration with frameworks like Virtualization

What are containers?

Containerization principles

Extending Container

Building on Containerization

Adoption of Swift

Open Source Community

Project goals

Project goals

- Grow the contributor community
  - Give us feedback!
  - Search the issues and PRs
  - File an issue before coding, describe your work
  - Review and understand what your AI wrote
  - Include clear reproduction steps for bug reports
  - Discussion topics for questions and design work

Project goals

- Encourage new projects built on top of Containerization

Project goals

- Encourage new projects built on top of Containerization
- Grow the container ecosystem through plugins

Areas of Investigation

Areas of Investigation

- Distro plugin for non-ephemeral Linux development environments

Areas of Investigation

- Distro plugin for non-ephemeral Linux development environments
- Kubernetes development environment using Containerization

Areas of Investigation

- Distro plugin for non-ephemeral Linux development environments
- Kubernetes development environment using Containerization
- Improving container image build performance by creating a Swift native builder

Areas of Investigation

- Distro plugin for non-ephemeral Linux development environments
- Kubernetes development environment using Containerization
- Improving container image build performance by creating a Swift native builder
- Ecosystem integration (orchestrators, dev containers)

Next Steps

- Try the tool
- Build something on top of Containerization, or extend container
- Join the community

github.com/apple/containerization

github.com/apple/container