Who I Think You Are

Software engineer, Sysadmin, etc who is...

- wanting to learn about namespaces and cgroups
- intereseted in containers and how they work
- loves turtles (optional)

Modern Linux Server with Containers

brandon.philips@coreos.com

Overview

Overview

- System Designs

Overview

- System Designs
- Namespaces

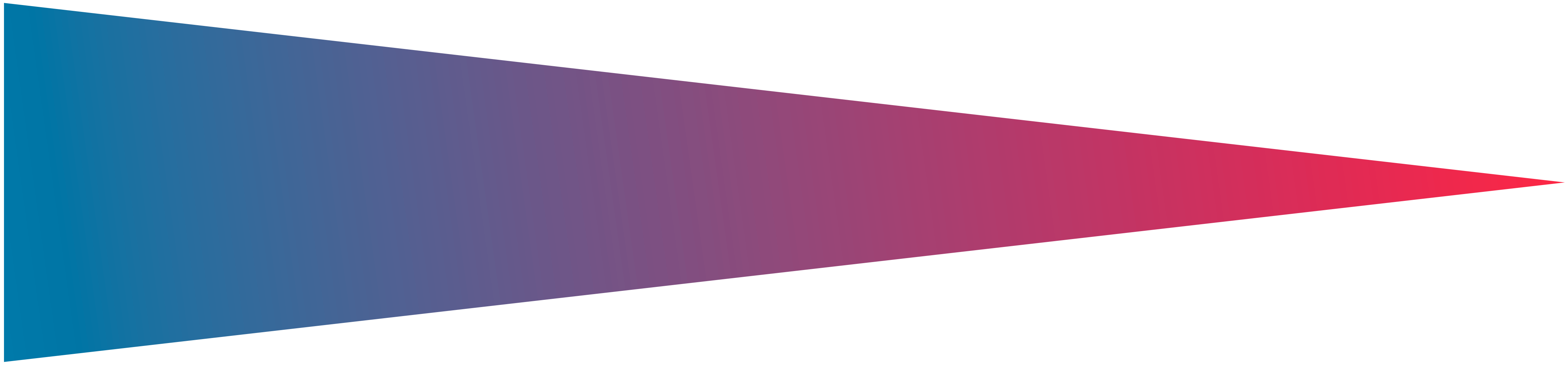
Overview

- System Designs
- Namespaces
- Cgroups

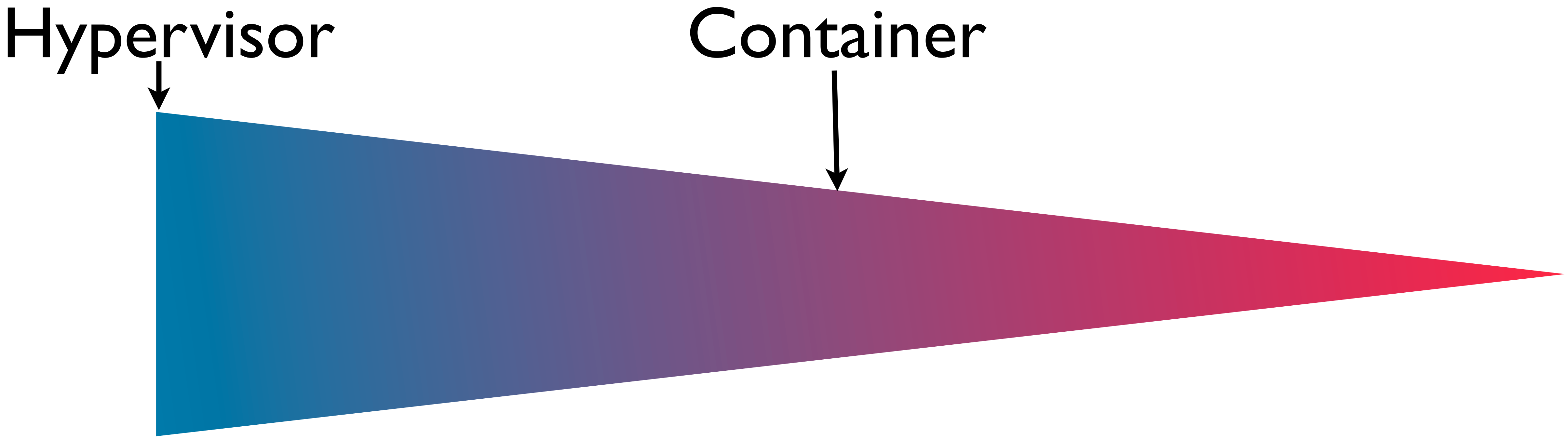
Overview

- System Designs
- Namespaces
- Cgroups
- Tooling

The Spectrum

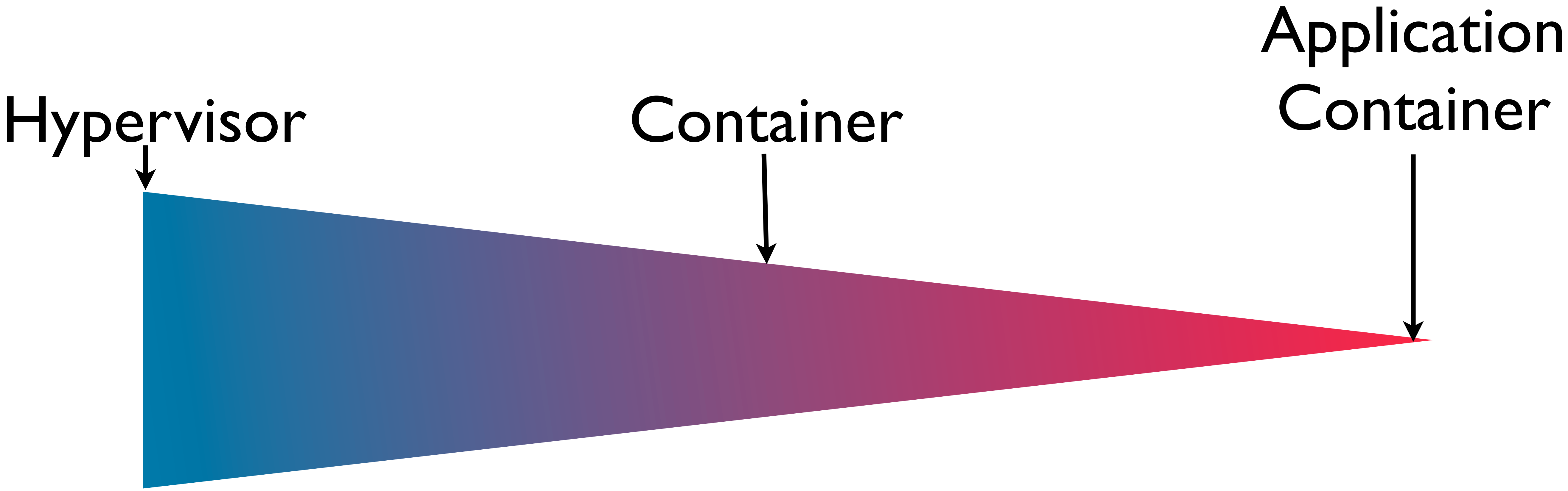


Hypervisor



Hypervisor

Container

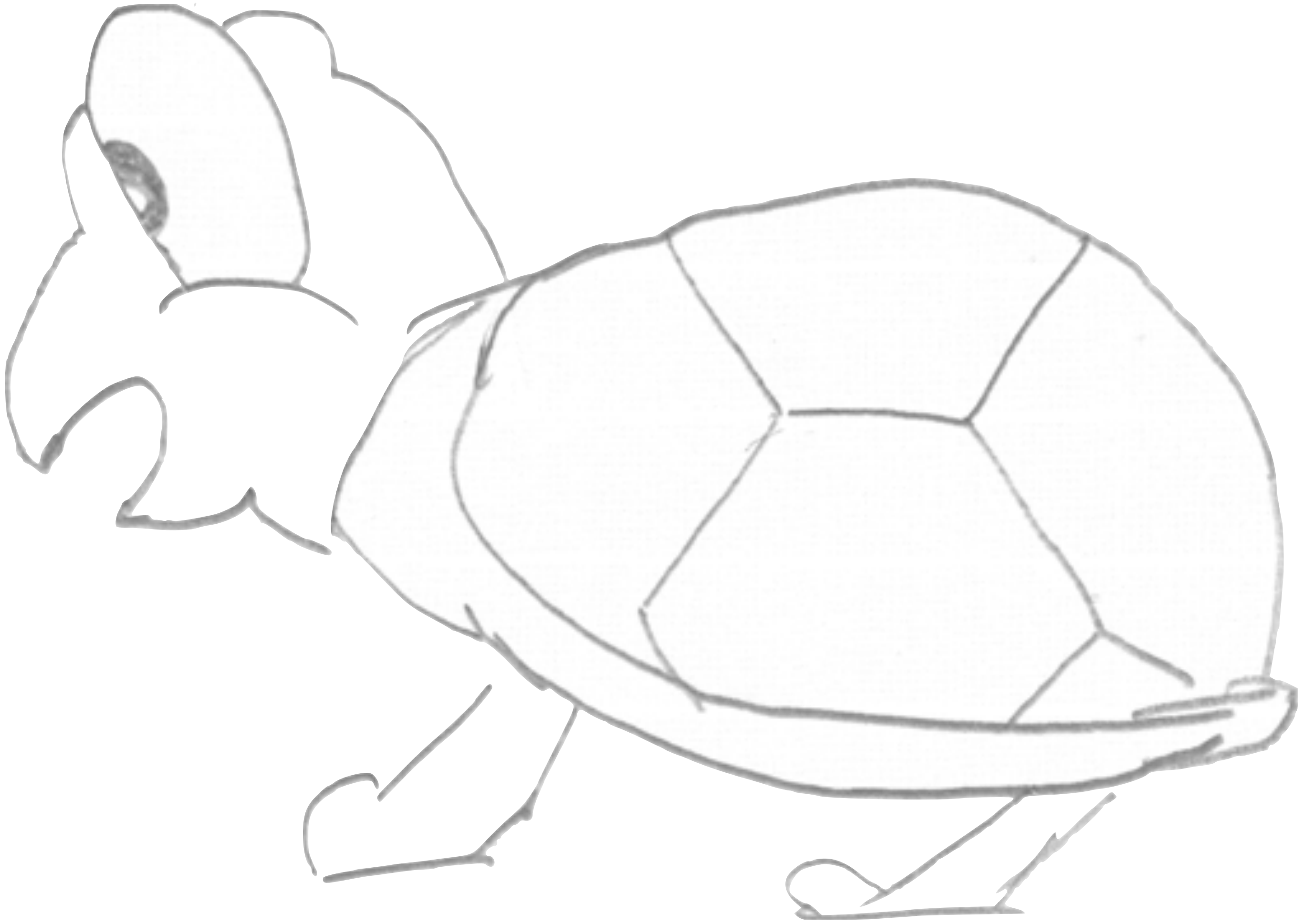


Hypervisor

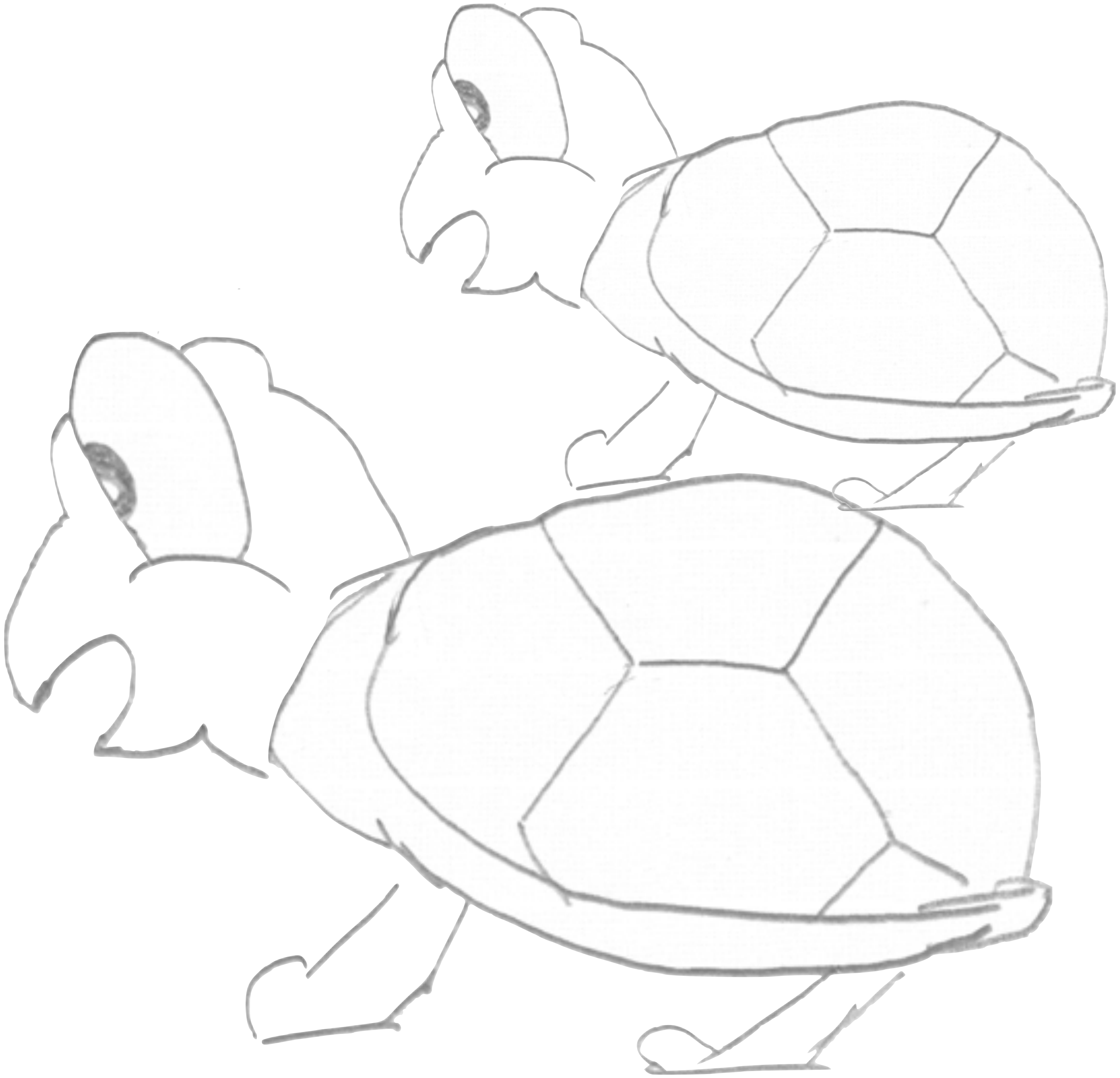
Container

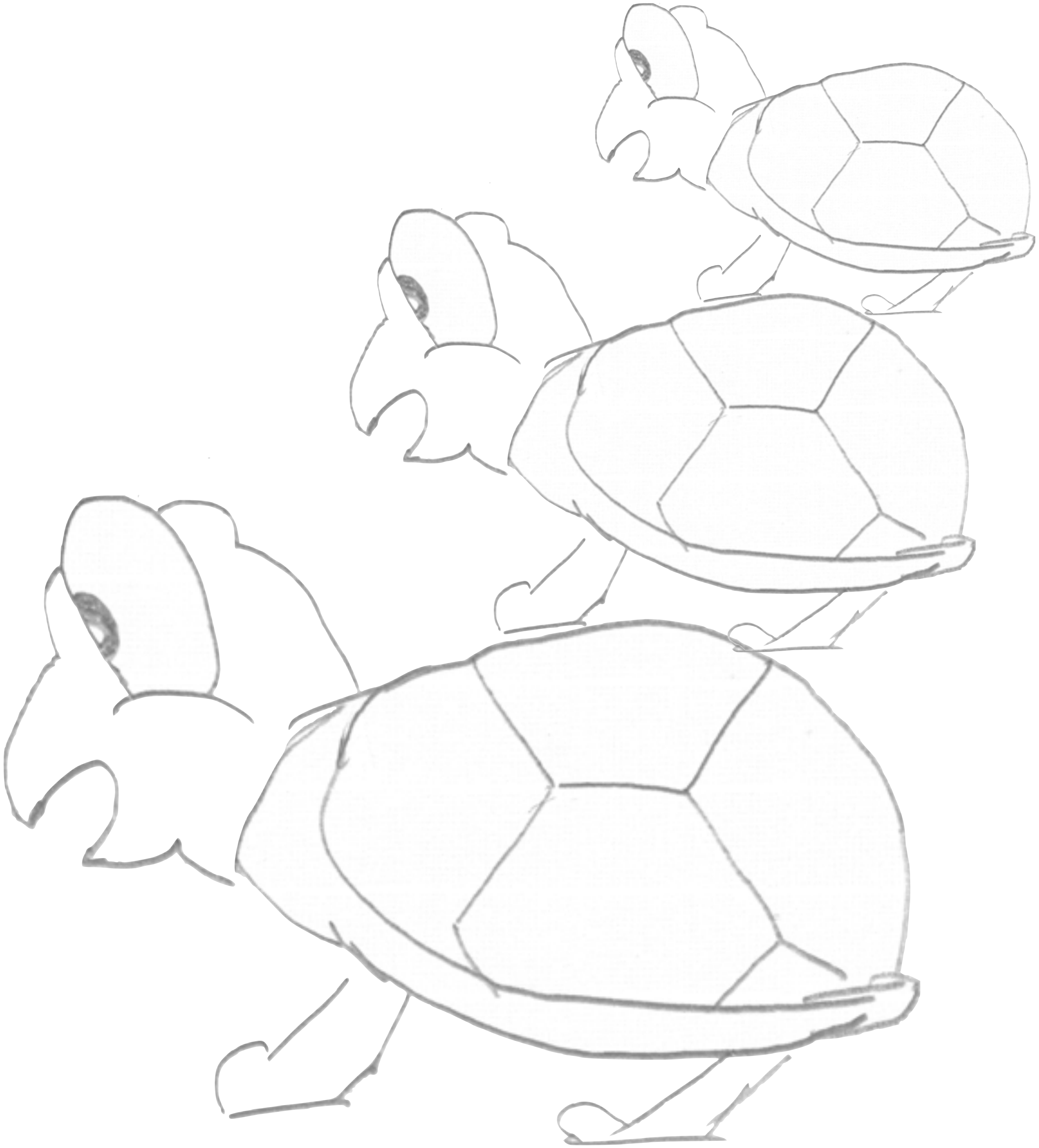
Application Container

WARNING

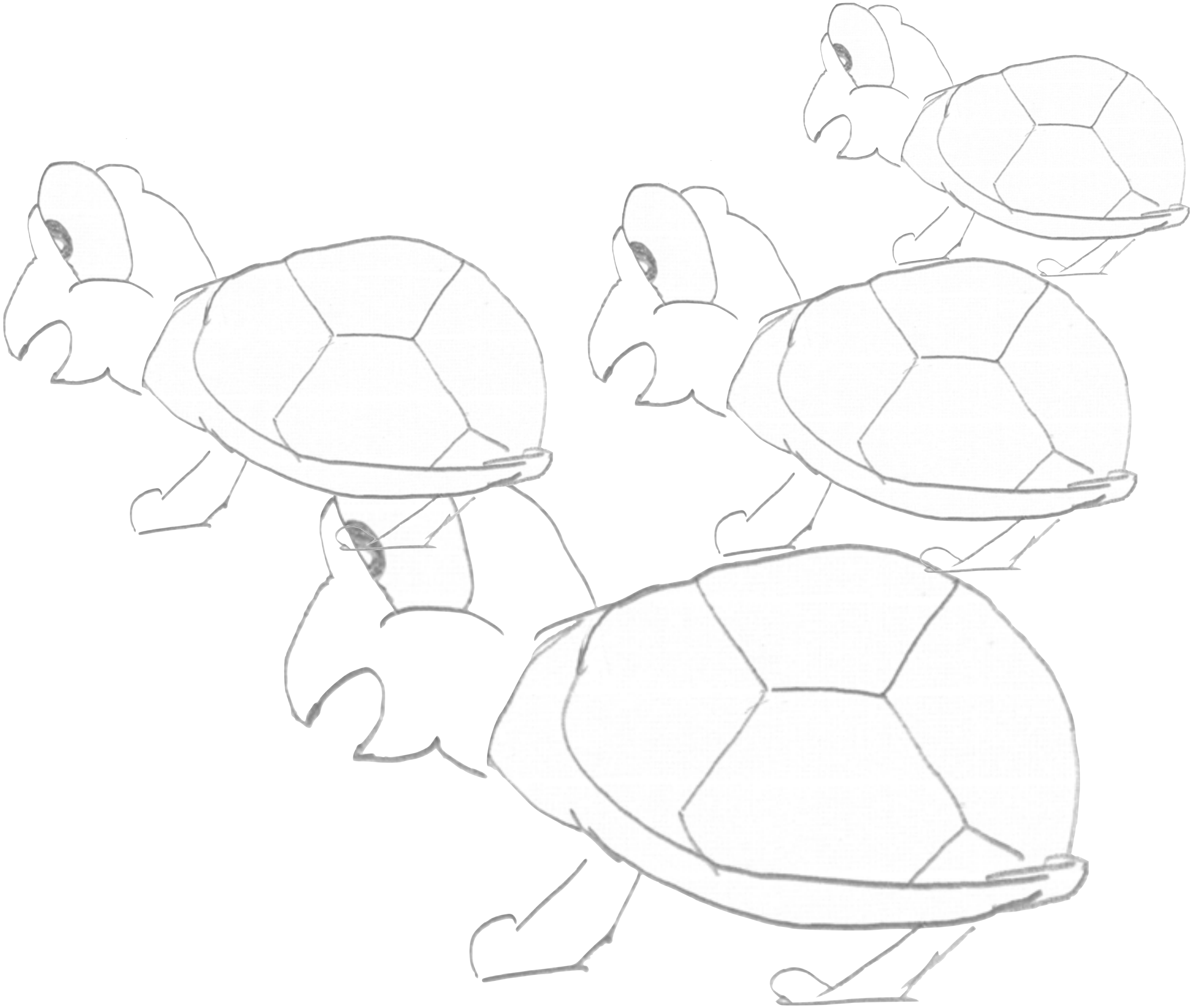


CoreOS

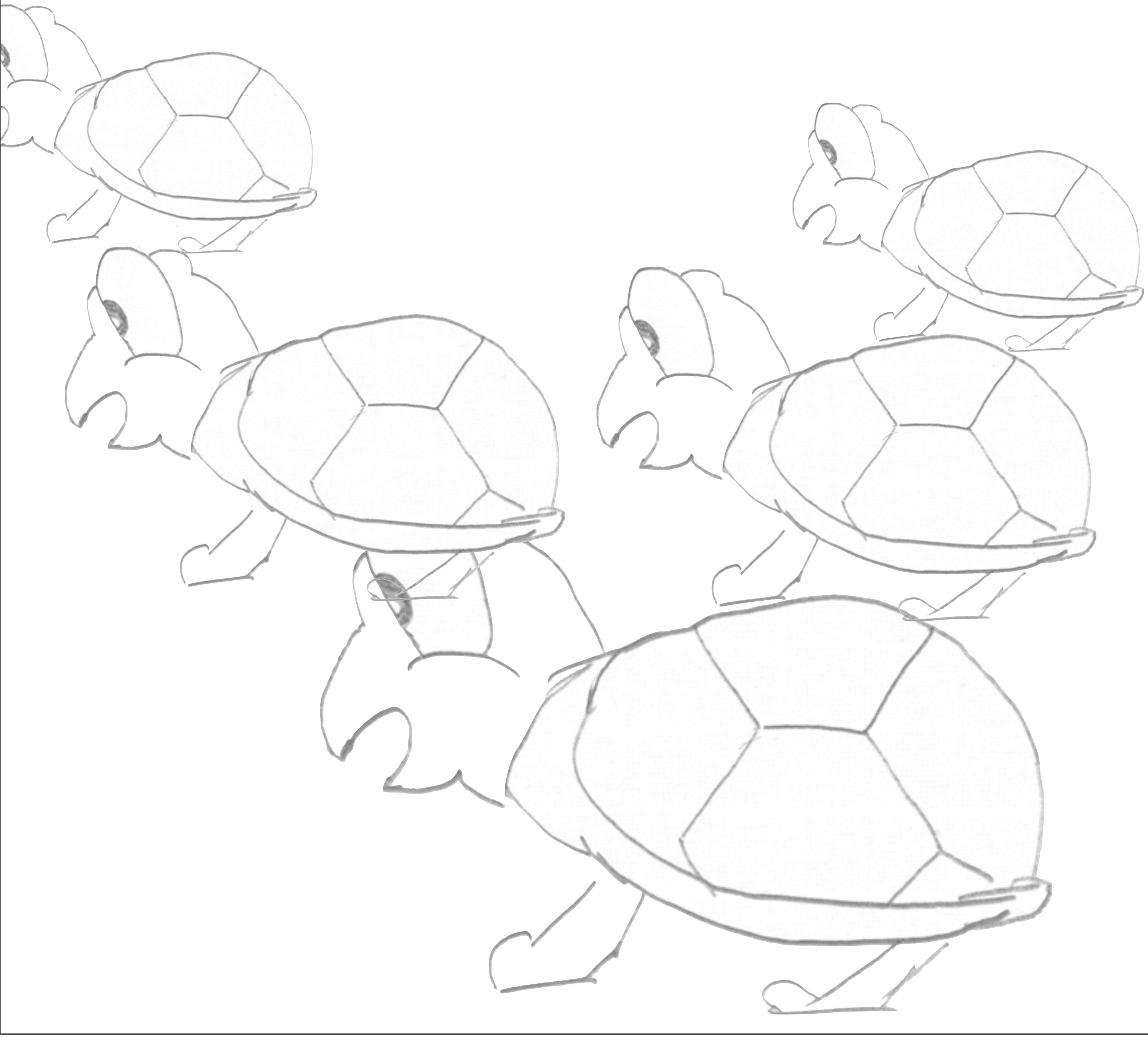




CoreOS

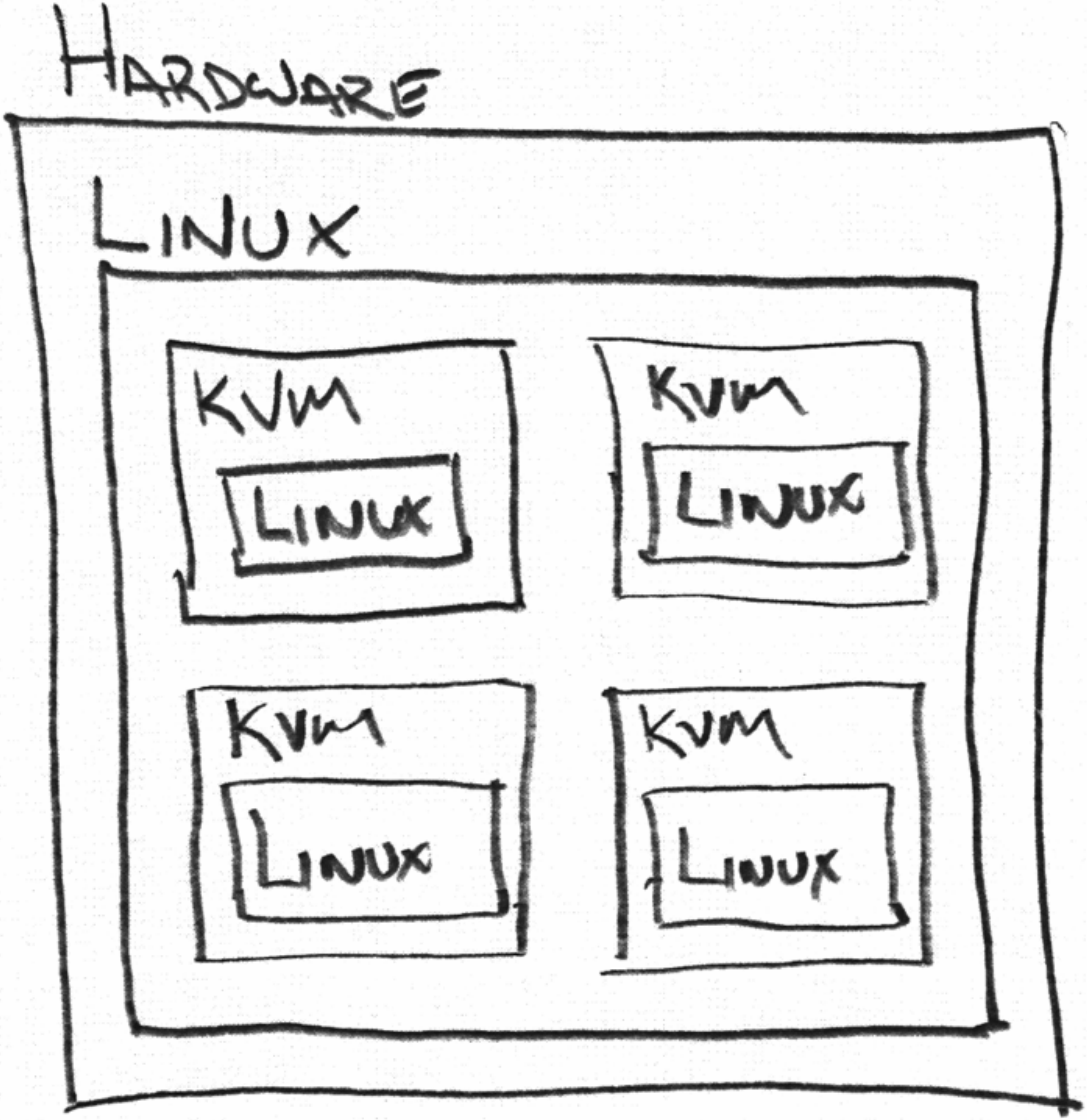


CoreOS



CoreOS

System Designs


HARDWARE
LINUX
KVM
LINUX
KVM
LINUX
KVM
LINUX
KVM
LINUX

Hypervisor

Hypervisor

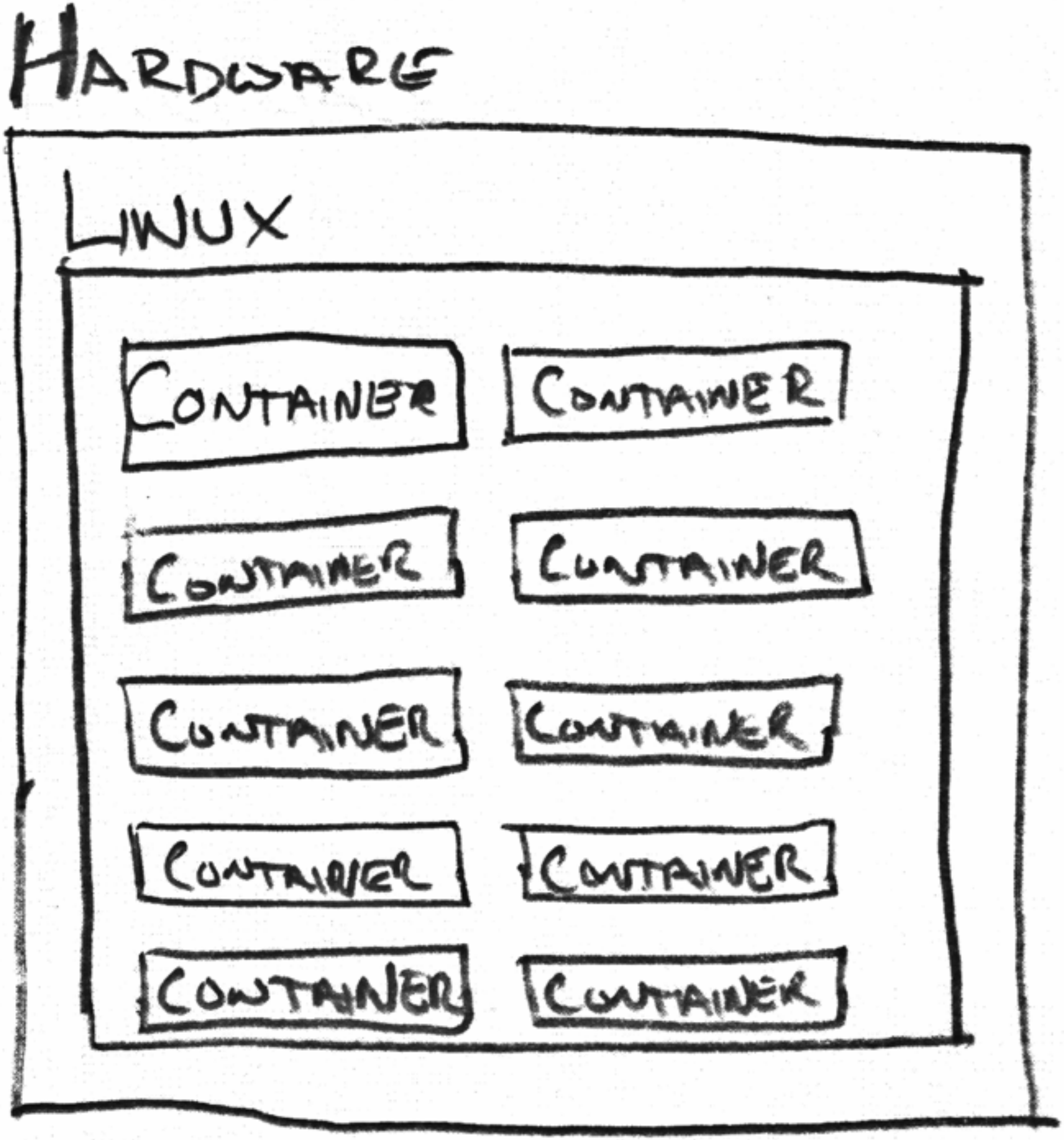
- Host provides full hardware environment

Hypervisor

- Host provides full hardware environment
- Block device, ethernet device, etc

Hypervisor

- Host provides full hardware environment
- Block device, ethernet device, etc
- Guests run a full kernel


HARDWARE
LINUX
CONTAINER
CONTAINER
CONTAINER
CONTAINER
CONTAINER
CONTAINER
CONTAINER
CONTAINER
CONTAINER
CONTAINER
CONTAINER
CONTAINER

Container

Container

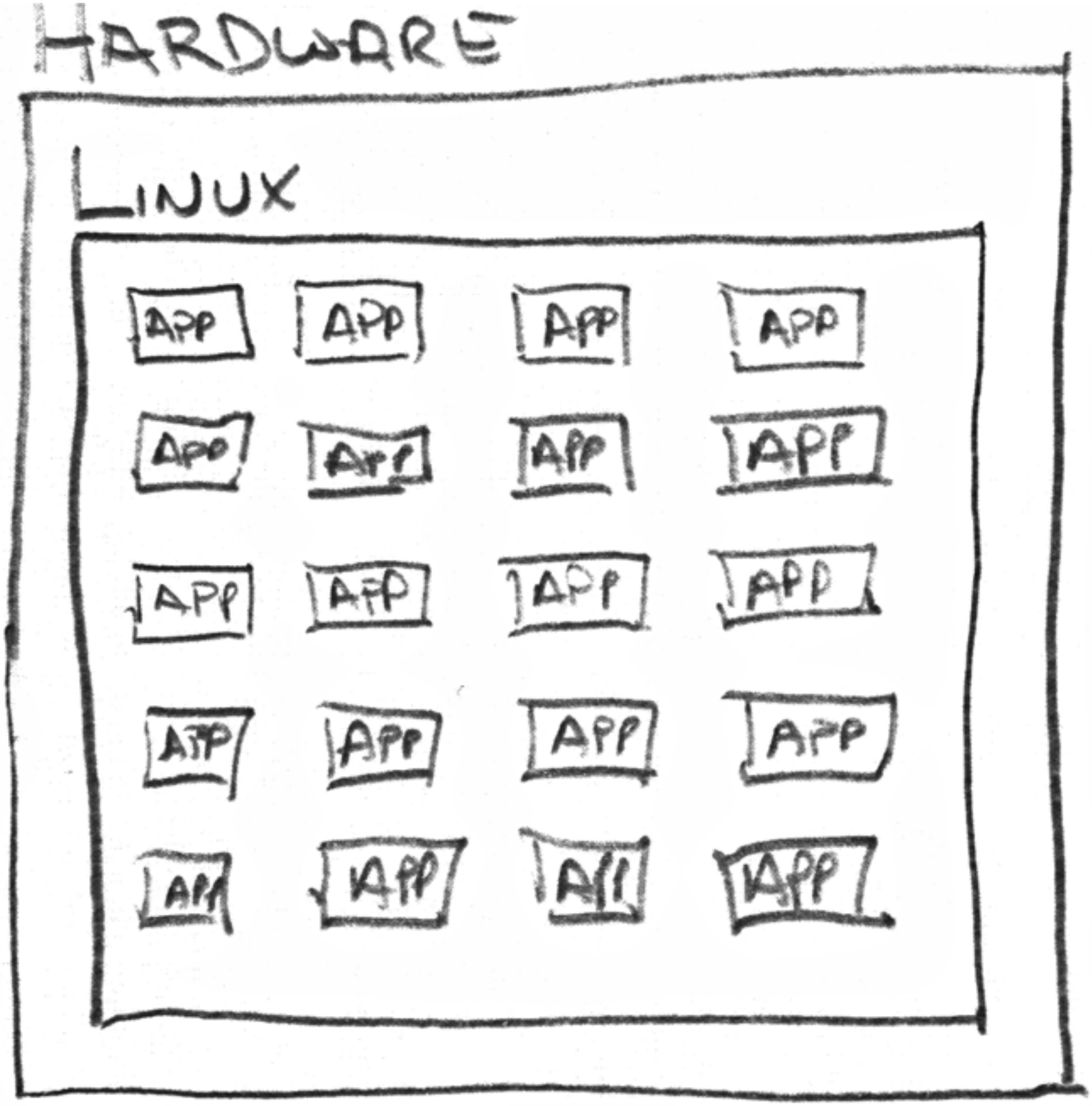
- Host provides Kernel

Container

- Host provides Kernel
- Filesystem, network interface, etc are already there

Container

- Host provides Kernel
- Filesystem, network interface, etc are already there
- Guest starts from /sbin/init


HARDWARE
LINUX
APP
APP
APP
APP
APP
APP
APP
APP
APP
APP
APP
APP
APP
APP
APP
APP
APP
APP
APP
APP

Application Container

Application Container

- Host provides Kernel

Core OS

Application Container

- Host provides Kernel
- User data, socket fd, etc are already there

Application Container

- Host provides Kernel
- User data, socket fd, etc are already there
- Starts from application not init

Core OS

Namespaces

Imagine: cool medieval castle photo

perhaps fog rolling in

Filesystem

Filesystem

- Read-only

Filesystem

- Read-only
- Shared

Filesystem

- Read-only
- Shared
- Slave

Filesystem

- Read-only
- Shared
- Slave
- Private

Read-only

Private bind mount

```
before:
after:
source/a-file
bind/a-file

mount -t tmpfs -o size=1M tmpfs source/mnt
before:
after:
source/mnt/tmpfs-file

mount -t tmpfs -o size=1M tmpfs bind/mnt2
before:
after:
bind/mnt2/mnt2-file
```

Shared bind mount

```
before:
after:
source/a-file
bind/a-file

mount -t tmpfs -o size=1M tmpfs source/mnt
before:
after:
source/mnt/tmpfs-file
bind/mnt/tmpfs-file

mount -t tmpfs -o size=1M tmpfs bind/mnt2
before:
after:
source/mnt2/mnt2-file
bind/mnt2/mnt2-file
```

Slave bind mount

```
before:
after:
source/a-file
bind/a-file

mount -t tmpfs -o size=1M tmpfs source/mnt
before:
after:
source/mnt/tmpfs-file
bind/mnt/tmpfs-file

mount -t tmpfs -o size=1M tmpfs bind/mnt2
before:
after:
bind/mnt2/mnt2-file
```

Patterns

- Mounting RO /usr inside a container
- Private /tmp per service
- Sharing data across containers via binds

Networking

Networking

- Root namespace

Networking

- Root namespace
- Bridging

Networking

- Root namespace
- Bridging
- Private namespace with socket activation

Root Namespace

- Full access to the machine interfaces

Root Namespace

Root Namespace

- **Advantages**

Root Namespace

- **Advantages**
- Fast

Root Namespace

- **Advantages**
- Fast
- Easy to get setup

Root Namespace

- **Advantages**
- Fast
- Easy to get setup
- Network looks normal to the container

Root Namespace

- **Advantages**
- Fast
- Easy to get setup
- Network looks normal to the container

Root Namespace

- **Advantages**
- Fast
- Easy to get setup
- Network looks normal to the container

- **Disadvatages**

Root Namespace

- **Advantages**
- Fast
- Easy to get setup
- Network looks normal to the container

- **Disadvatages**
- No separation of concerns

Root Namespace

- **Advantages**
- Fast
- Easy to get setup
- Network looks normal to the container

- **Disadvatages**
- No separation of concerns
- Container has full control

Network Bridges

Network Bridges

- Create a bridge, like a virtual switch

Network Bridges

- Create a bridge, like a virtual switch
- Create container namespace and add interface

Network Bridges

- Create a bridge, like a virtual switch
- Create container namespace and add interface
- Attach container interface to bridge

Network Bridges

Network Bridges

- **Advantages**

Network Bridges

- **Advantages**
- More complex to get setup

Network Bridges

- **Advantages**
- More complex to get setup
- Network looks normal to the container

Network Bridges

- **Advantages**
- More complex to get setup
- Network looks normal to the container

Network Bridges

- **Advantages**
- More complex to get setup
- Network looks normal to the container

Network Bridges

- **Advantages**
- More complex to get setup
- Network looks normal to the container

- **Disadvantages**

Network Bridges

- **Advantages**
- More complex to get setup
- Network looks normal to the container

- **Disadvantages**
- Less speed

Network Bridges

- **Advantages**
- More complex to get setup
- Network looks normal to the container

- **Disadvantages**
- Less speed
- NAT to the internet

Network Bridges

- **Advantages**
- More complex to get setup
- Network looks normal to the container

- **Disadvantages**
- Less speed
- NAT to the internet
- iptables to expose public socket

Socket Activation

Socket Activation

- No interface

Socket Activation

- No interface
- Sockets are passed via stdin (inetd)

Socket Activation

- No interface
- Sockets are passed via stdin (inetd)
- systemd style listen fd API

inetd style

inetd style

- **Advantages**

inetd style

- **Advantages**
- Fast and isolated

inetd style

- **Advantages**
- Fast and isolated
- Simple and well understood

inetd style

- **Advantages**
- Fast and isolated
- Simple and well understood
- Support from existing daemons like ssh

inetd style

- **Advantages**
- Fast and isolated
- Simple and well understood
- Support from existing daemons like ssh
- No process running until needed

inetd style

- **Advantages**
- Fast and isolated
- Simple and well understood
- Support from existing daemons like ssh
- No process running until needed

- **Disadvantages**

inetd style

- **Advantages**
- Fast and isolated
- Simple and well understood
- Support from existing daemons like ssh
- No process running until needed

- **Disadvantages**
- One process per client (scaling problems!)

listen fd style

listen fd style

- **Advantages**

listen fd style

- **Advantages**
- Fast and isolated

listen fd style

- **Advantages**
- Fast and isolated
- Only one process needed per service

listen fd style

- **Advantages**
- Fast and isolated
- Only one process needed per service
- No process running until needed

listen fd style

- **Advantages**
- Fast and isolated
- Only one process needed per service
- No process running until needed

listen fd style

- **Advantages**
- Fast and isolated
- Only one process needed per service
- No process running until needed

- **Disadvantages**

listen fd style

- **Advantages**
- Fast and isolated
- Only one process needed per service
- No process running until needed

- **Disadvantages**
- Patches required to daemons

Process Namespace

- PID 1 is something else outside the namespace

All the Rest

Cgroups

CoreOS

Imagine: an accountant’s overflowing desk
perhaps hands on head in dispair

Block I/O

- **Limit**: Weight from 10 to1000
- **Limit**: Bandwidth limits R/W
- **Metrics:** iops serviced, waiting and queued

CPU

- **Limit**: Shares system 1024 is half of 2048
- **Metrics:** cpuacct.stats user and system

Memory

- **Limit**: Total RSS memory limit
- **Metrics**: swap, total rss, # page ins/outs

Tooling

Core OS

docker

nspawn

nsenter

/sys/fs/cgroup

systemd units

systemd-cgtop

Recap

Recap

- Containers are built on namespaces and cgroups

Recap

- Containers are built on namespaces and cgroups
- Namespaces provide isolation similar to hypervisors

Recap

- Containers are built on namespaces and cgroups
- Namespaces provide isolation similar to hypervisors
- Cgroups provide resource limiting and accounting

Recap

- Containers are built on namespaces and cgroups
- Namespaces provide isolation similar to hypervisors
- Cgroups provide resource limiting and accounting
- These tools can be mixed to create hybrids

Future

CoreOS

Thanks!

@BrandonPhilips
@CoreOSLinux