THE ROUTE

To Rootless Containers

@DOCTOR_JULZ

IBMer

Garden (CF Containers)
PM / Project Lead

???

@EDKING2

Pivot

Garden (CF Containers)
Anchor / Tech Lead

???

THE ROUTE

To Rootless Containers

THE ROUTE

To Rootless Containers

THE ROUTE

Rootless Containers

THE ROUTE

Rootless Containers

JULZ AND TED'S ROOTLESS ADVENTURE

WHY DO WE CARE?

1. CONTAINER SECURITY

2. ROOTLESS!

3.

WHY DO WE CARE?

WHY DO WE CARE?

AND SO SHOULD YOU

WHY DO WE CARE?

AND SO SHOULD YOU

CLOUD FOUNDRY

WHY DO WE CARE?

AND SO SHOULD YOU

CLOUD FOUNDRY

- **Platform as a Service**
- **Heroku-like**
- **Very very popular with big companies!**

WHY DO WE CARE?

AND SO SHOULD YOU

CLOUD FOUNDRY

- Public Cloud
- Multi-tenant
- Allows running Docker Images

WHY DO WE CARE?

AND S YOU

CLOUD F

oud

ant

- Allows running Docker Images

WHY DO WE CARE?

A U

CL

r

images

WHY DO WE CARE?

AND SO SHOULD YOU

- Worst case scenario!
- Bleeding edge of container security

CONTAINER SECURITY

CONTAINER SECURITY

"THE GREATEST TRICK CONTAINERS EVER PULLED WAS CONVINCING THE WORLD THEY EXIST"

CONTAINER SECURITY

WHAT IS A CONTAINER?

CONTAINER SECURITY

WHAT IS A CONTAINER?

Jar Files*

(and like a billion other things)

WHAT IS A CONTAINER?

Write Once

WHAT IS A CONTAINER?

"Run Anywhere"

CONTAINER SECURITY

WHAT IS A CONTAINER?

But Isolation!

WHAT IS A CONTAINER?

Namespaces

CONTAINER SECURITY

WHAT IS A CONTAINER?

(Isolation)

CONTAINER SECURITY

WHAT IS A CONTAINER?

Cgroups

WHAT IS A CONTAINER?

(Fair Sharing)

WHAT IS A CONTAINER?

Namespaces + Cgroups, Yay!

WHAT IS A CONTAINER?

Docker

CONTAINER SECURITY

WHAT IS A CONTAINER?

Encapsulation

WHAT IS A CONTAINER?

“Containers”

WHAT IS A CONTAINER?

1. Isolation
2. Resource Sharing
3. Encapsulation

WHAT IS A CONTAINER?

1. Isolation

“Linux Container”

2. Resource Sharing

3. Encapsulation

WHAT IS A CONTAINER?

1. Isolation
2. Resource Sharing

“Linux Container”

3. Encapsulation

“Container”

WHAT IS A CONTAINER?

ISOLATION

WHAT IS A CONTAINER?

ISOLATION

Namespaces

WHAT IS A CONTAINER?

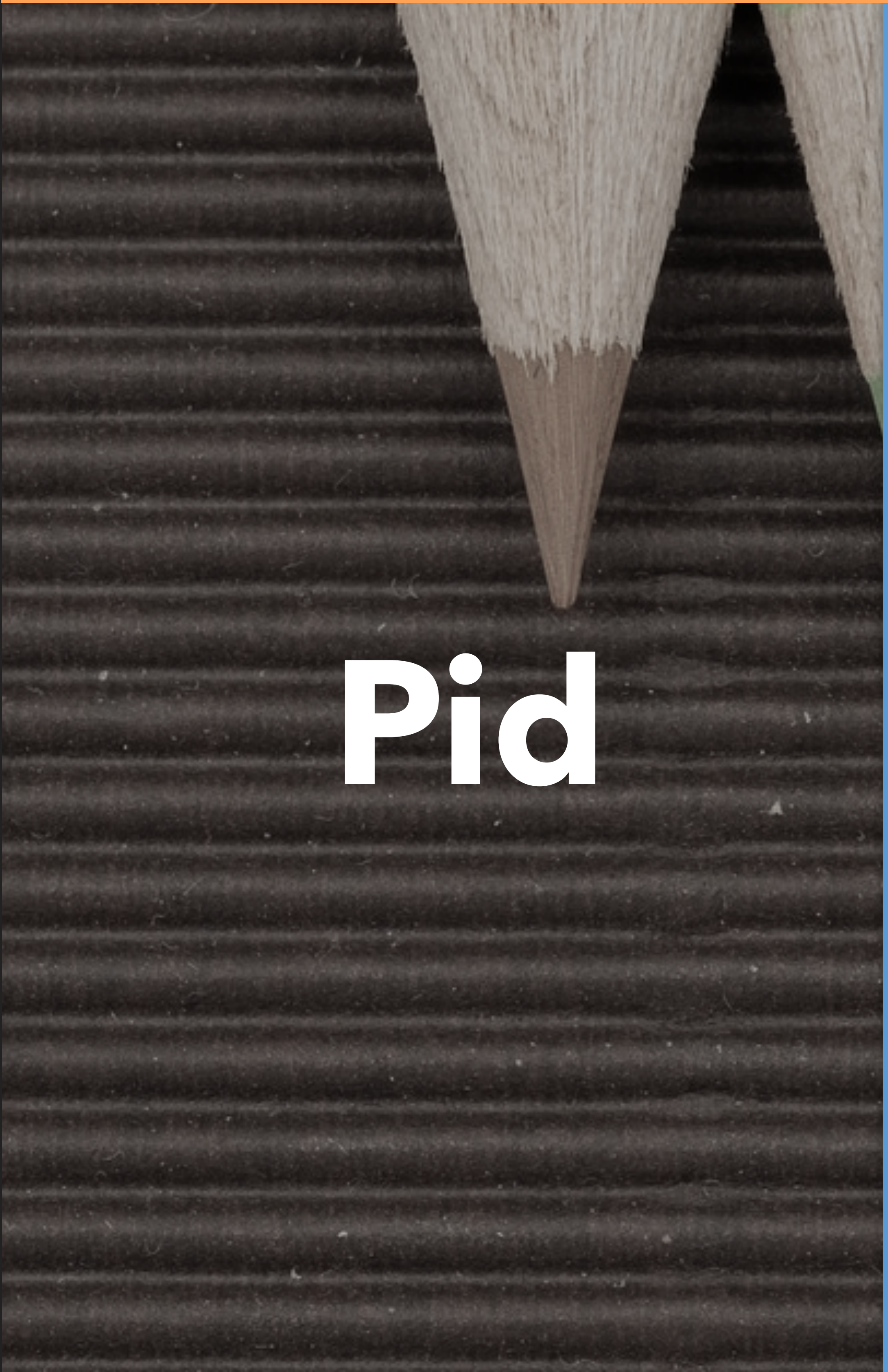
ISOLATION

Namespaces

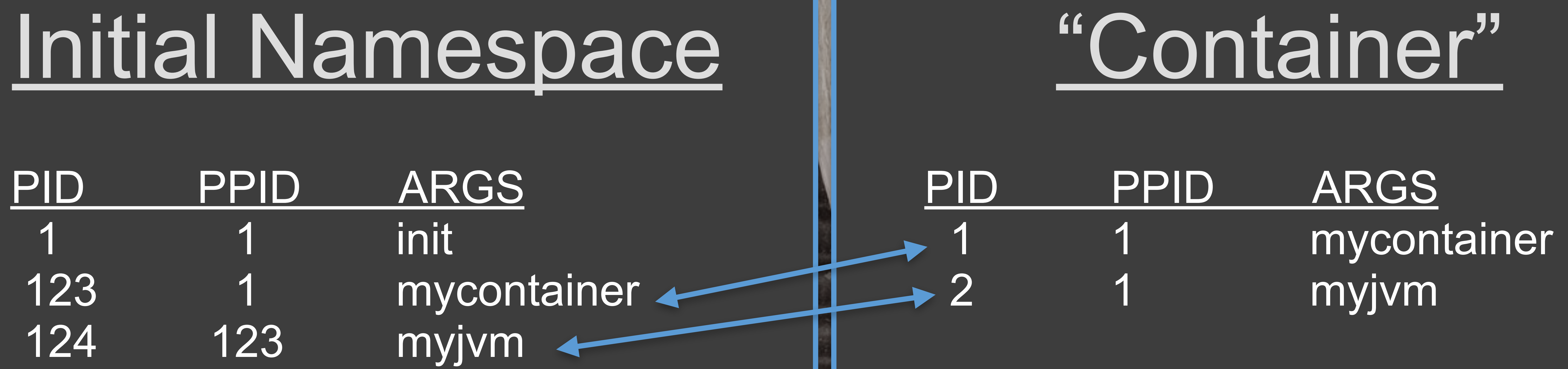
- Pid
- Mnt
- Net
- IPC
- UTC
- User
- Cgroup

WHAT IS A CONTAINER?

EXAMPLE: PID NS



Pid



Initial Namespace

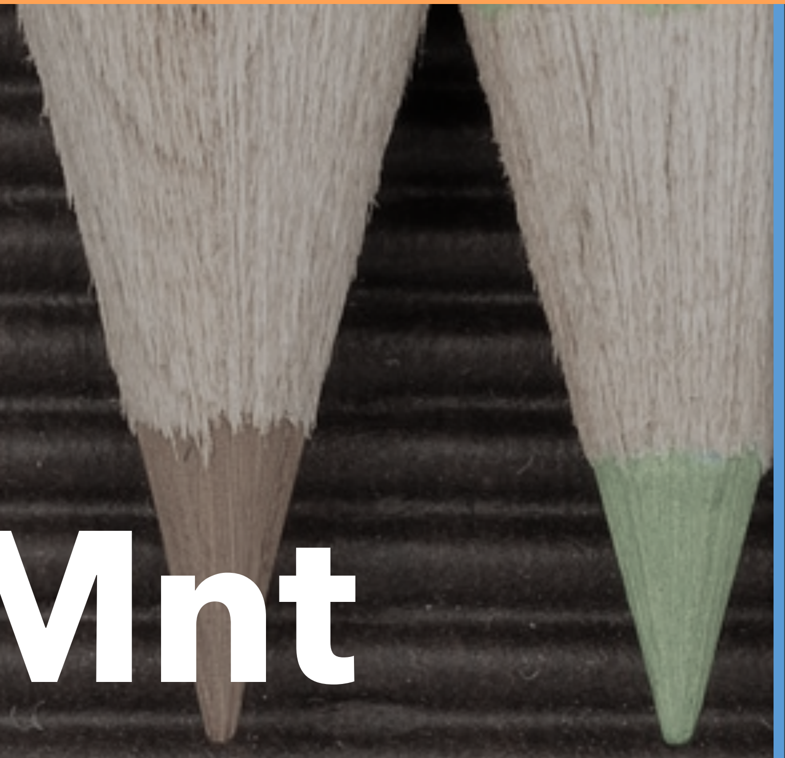
PID	PPID	ARGS
1	1	init
123	1	mycontainer
124	123	myjvm

“Container”

PID	PPID	ARGS
1	1	mycontainer
2	1	myjvm

WHAT IS A CONTAINER?

EXAMPLE: MOUNT NS



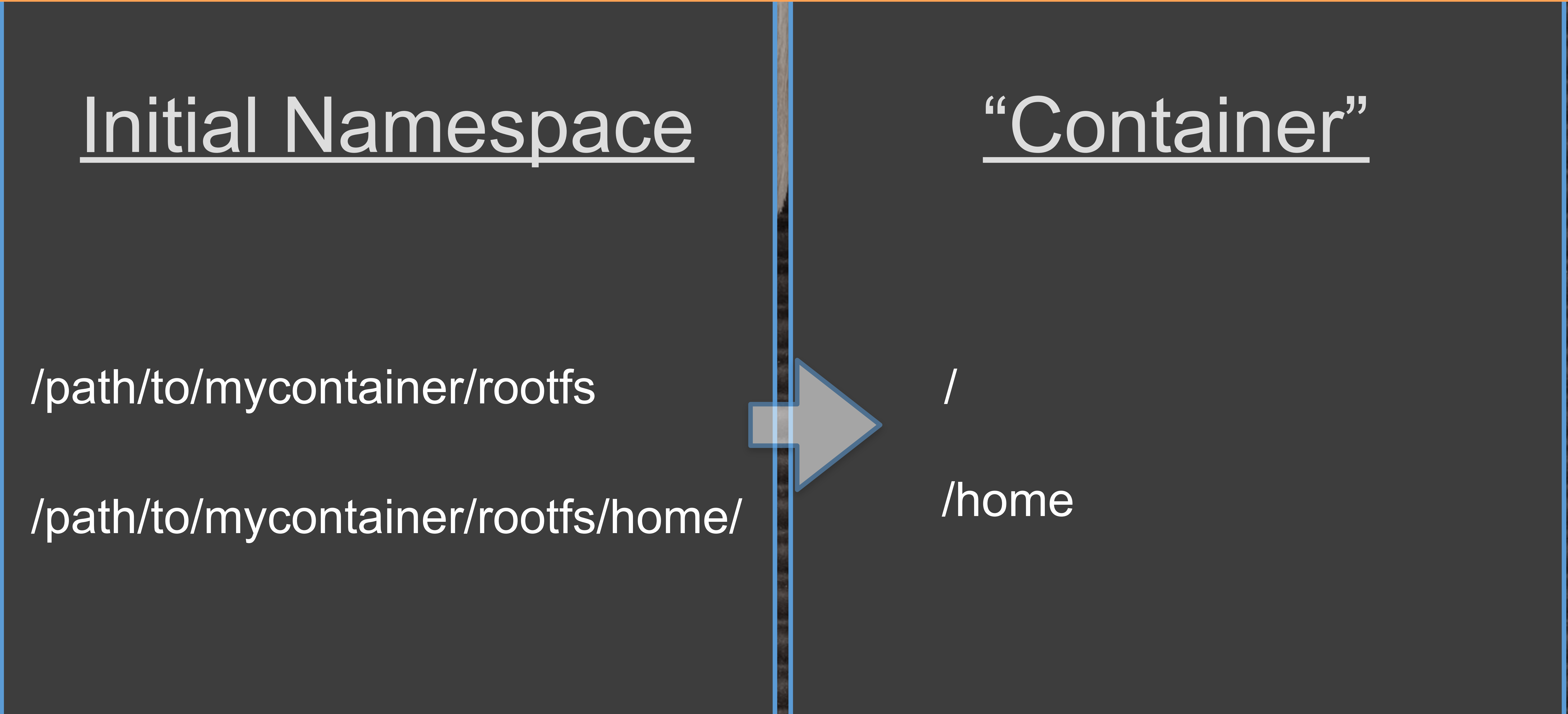
Mnt

(Namespace)

+

Pivot_Root

(Syscall)



Initial Namespace

/path/to/mycontainer/rootfs

/path/to/mycontainer/rootfs/home/

“Container”

/

/home

WHAT IS A CONTAINER?

ISOLATION

Pid + Mount + Net + IPC + User + UTC + Cgroup

Namespaces

WHAT IS A CONTAINER?

ISOLATION

That’s some nice isolation you got there

Be a shame if someone broke out of it …

WHAT IS A CONTAINER?

SECURITY ONION

WHAT IS A CONTAINER?

SECURITY ONION

- Capability Dropping
- Seccomp
- AppArmor

WHAT IS A CONTAINER?

SECURITY ONION: CAPS

WHAT IS A CONTAINER?

SECURITY ONION: CAPS

Previously: all-powerful “root” user

WHAT IS A CONTAINER?

SECURITY ONION: CAPS

Nowadays: split in to multiple "capabilities"

WHAT IS A CONTAINER?

SECURITY ONION: CAPS

- **CAP_SET_UID - Change UID**
- **CAP_NET_BIND_SERVICE - Listen on privileged ports**
- **CAP_KILL - Send signals to any process**
- **CAP_CHOWN - chown any file**
- **CAP_SYS_ADMIN - Do all the things?!**

WHAT IS A CONTAINER?

SECURITY ONION: SECCOMP

- “Secure Computing Mode”
- Basically, limit system calls a process can make
- Pretty great, exploits in those don’t hurt you any more

WHAT IS A CONTAINER?

SECURITY ONION: APPARMOR

- “Mandatory Access Control”
- See also: SELinux
- Example Rule:
  deny @{PROC}/* w

WHAT IS A CONTAINER?

SECURITY ONION

- Capability Dropping
- Seccomp
- AppArmor

I GET KNOCKED DOWN (BUT I GET UP AGAIN)

- **CVE-2016-9962**: runc fd traversal: User Namespaces, Capability Dropping, AppArmor
- **CVE-2017-16539:** SCSI MICDROP - User Namespaces, AppArmor
- **CVE-2017-16995**: eBPF verifier vulnerability - Capability Dropping (sometimes), Seccomp

WHAT IS A CONTAINER?

Isolation

2. Resource Sharing

3. Encapsulation

WHAT IS A CONTAINER?

✓ Isolation

2. Resource Sharing

3. Encapsulation

WHAT IS A CONTAINER?

RESOURCE SHARING

Cgroups

WHAT IS A CONTAINER?

RESOURCE SHARING

CPU*

* CPU, CPUSet, CPUAcct

Memory

Blkio

Pids

Devices

Freezer

Net*

* Net_prio, Net_cls

Cgroups

WHAT IS A CONTAINER?

RESOURCE SHARING

Disk Quotas

WHAT IS A CONTAINER?

Disk Quotas

WHAT IS A CONTAINER?

Disk Quotas
(more later)

WHAT IS A CONTAINER?

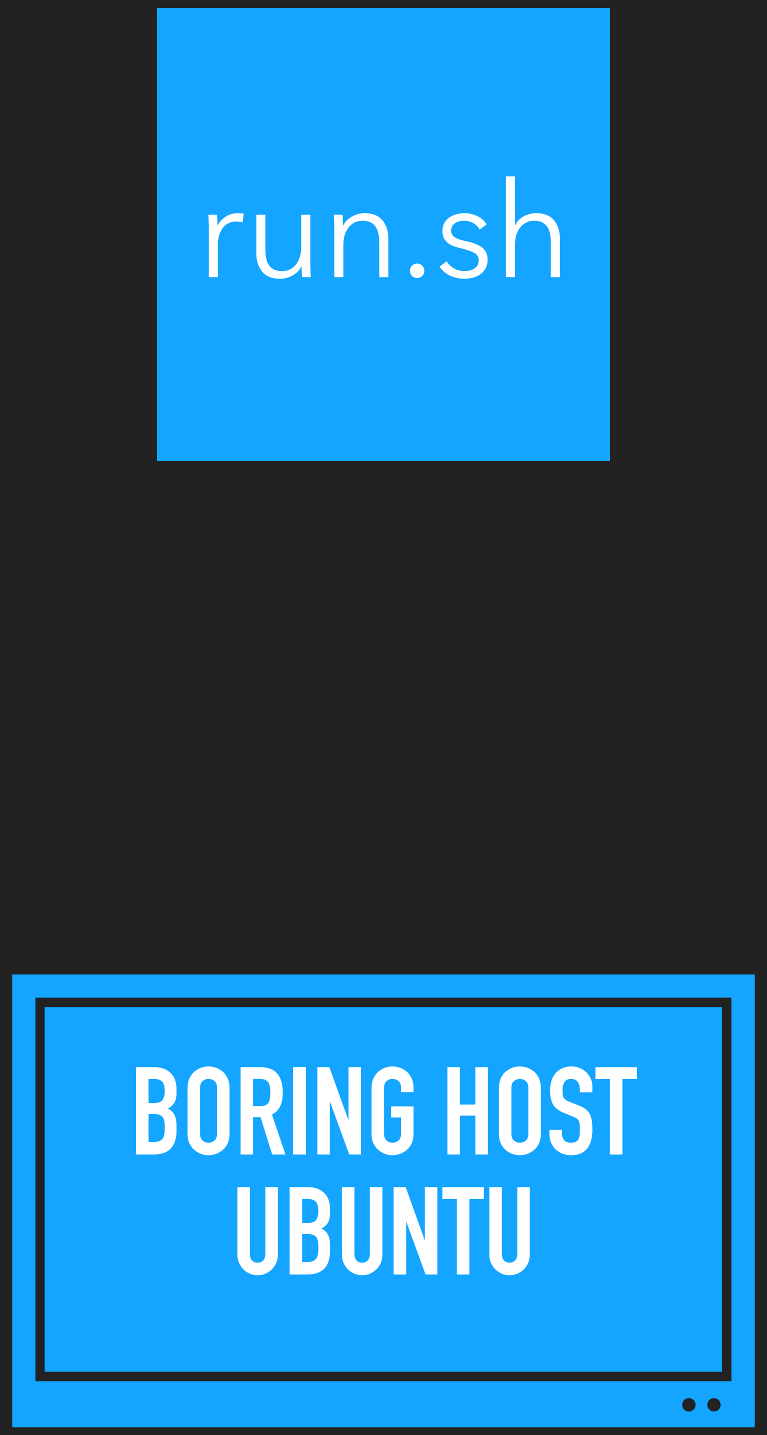
✓ Isolation

✓ Resource Sharing

3. Encapsulation

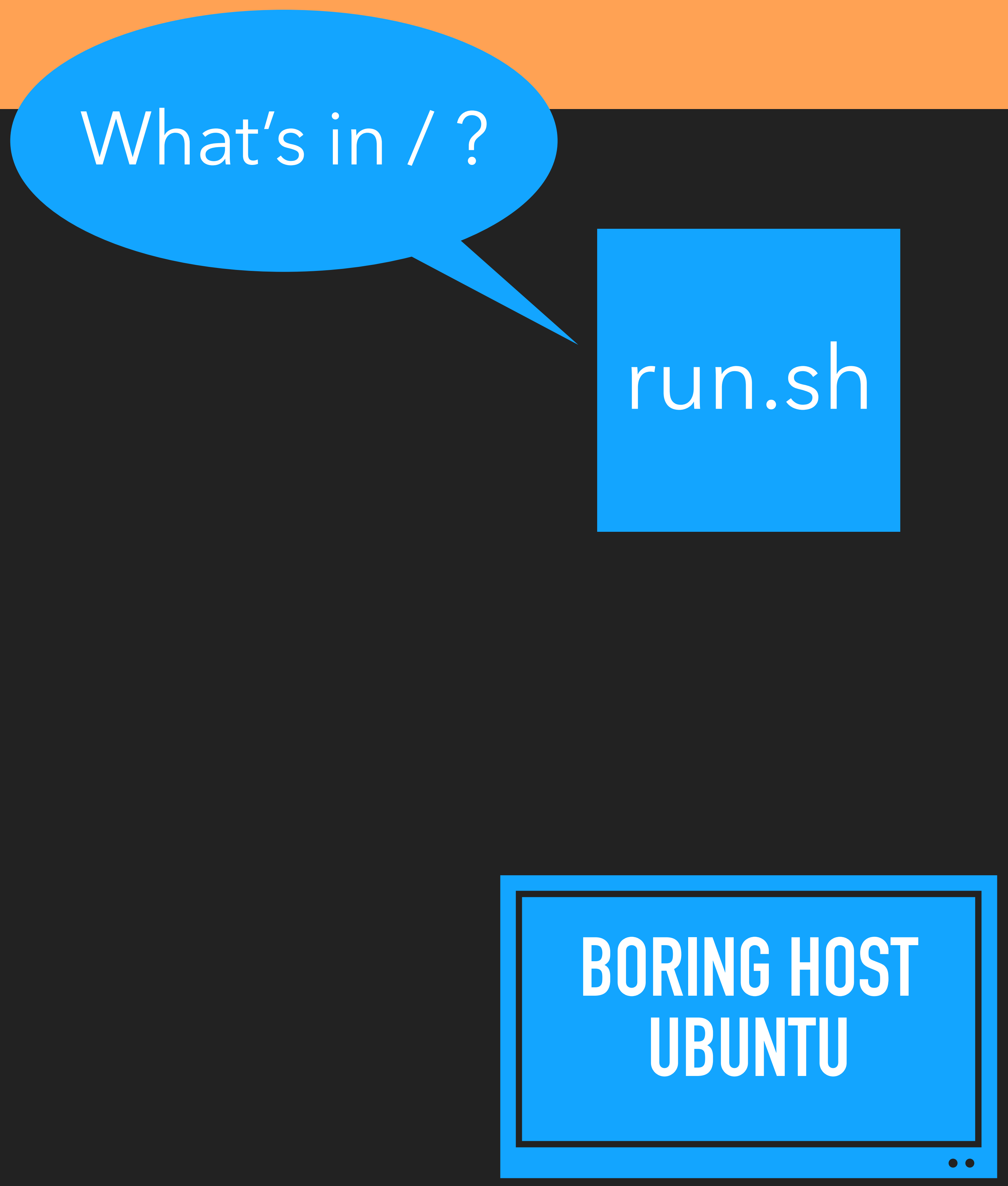
WHAT IS A CONTAINER?

ENCAPSULATION: PIVOT_ROOT



WHAT IS A CONTAINER?

ENCAPSULATION: PIVOT_ROOT



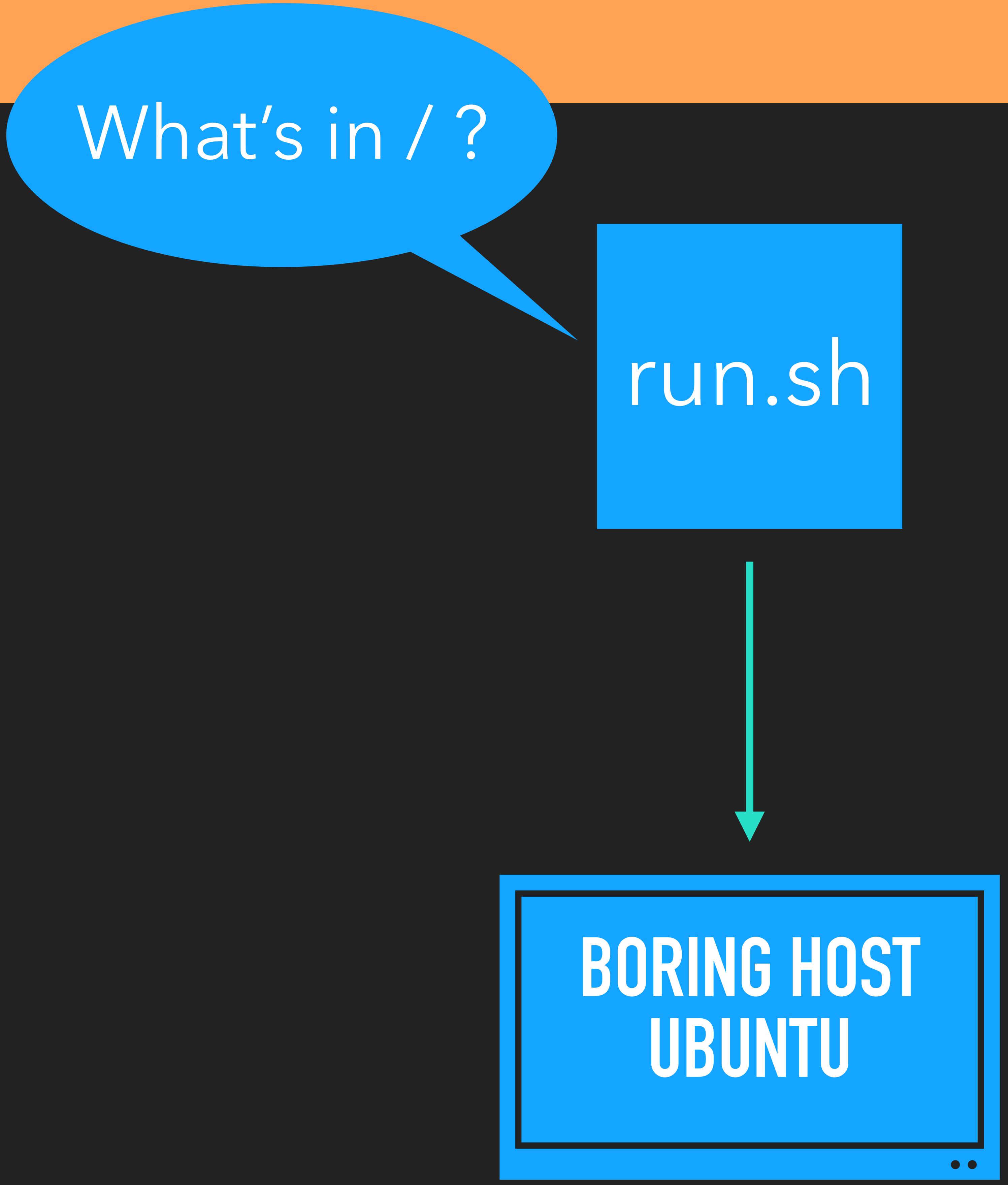
What's in / ?

run.sh

BORING HOST
UBUNTU

WHAT IS A CONTAINER?

ENCAPSULATION: PIVOT_ROOT



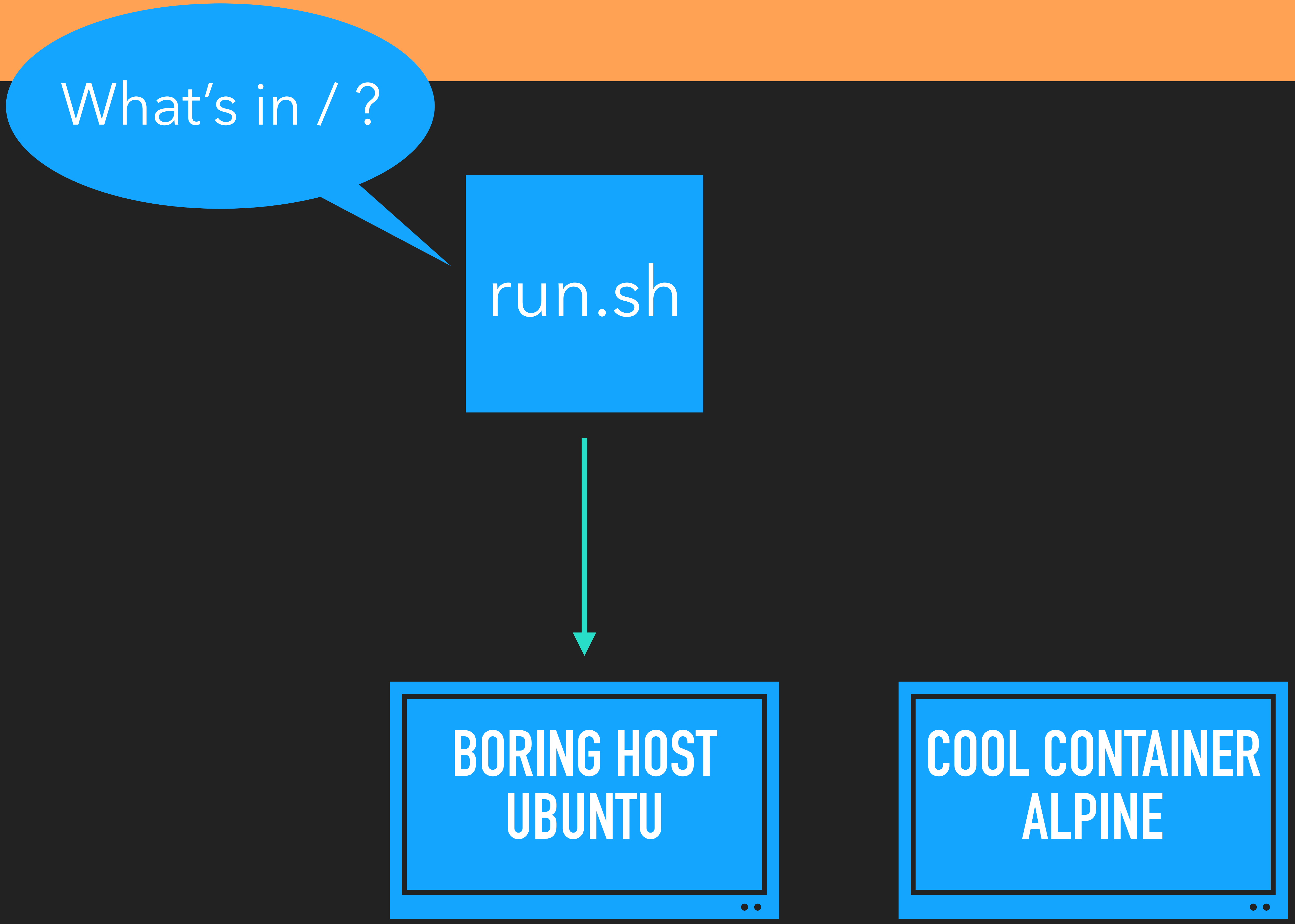
What's in / ?

run.sh

BORING HOST UBUNTU

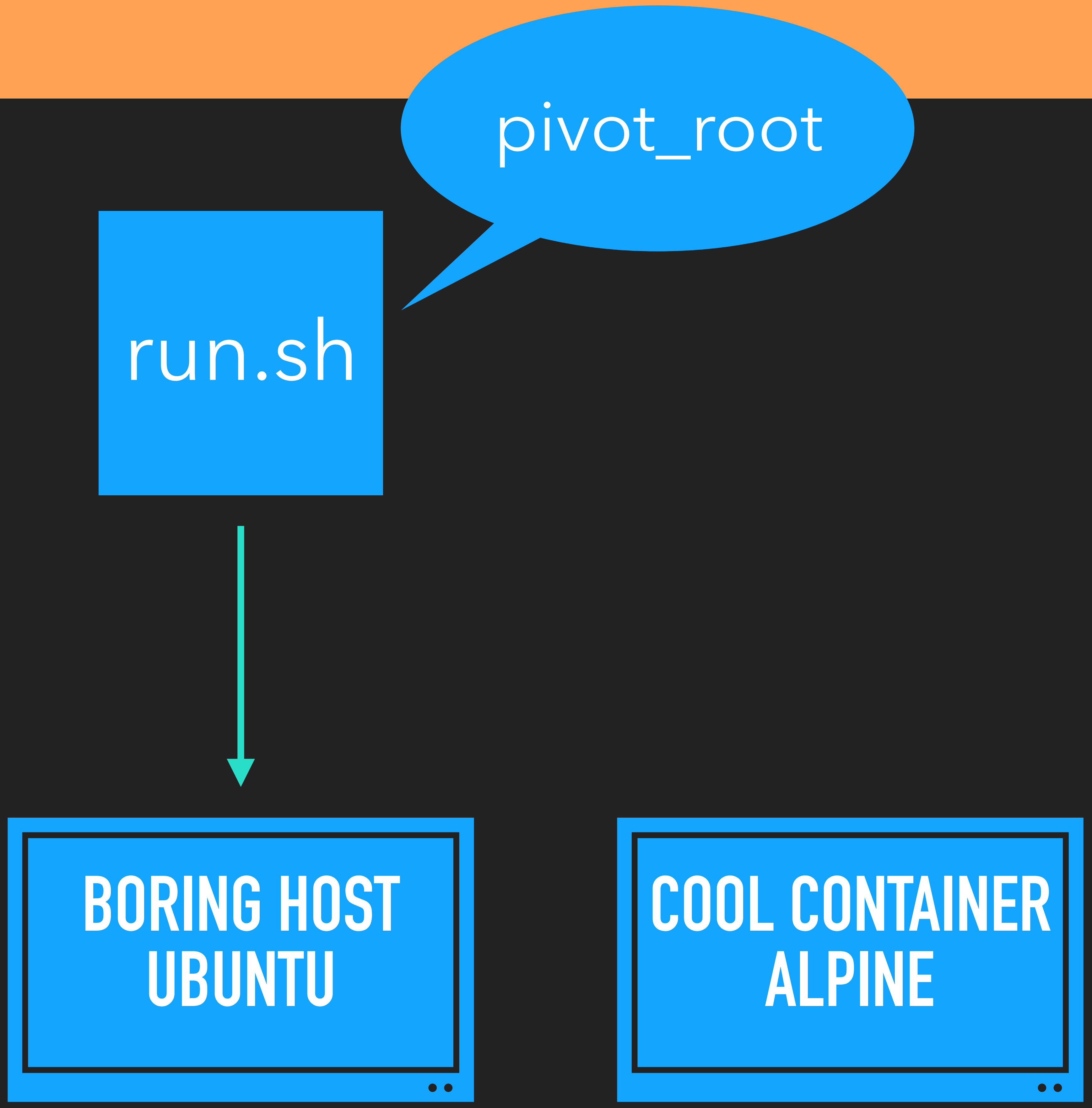
WHAT IS A CONTAINER?

ENCAPSULATION: PIVOT_ROOT



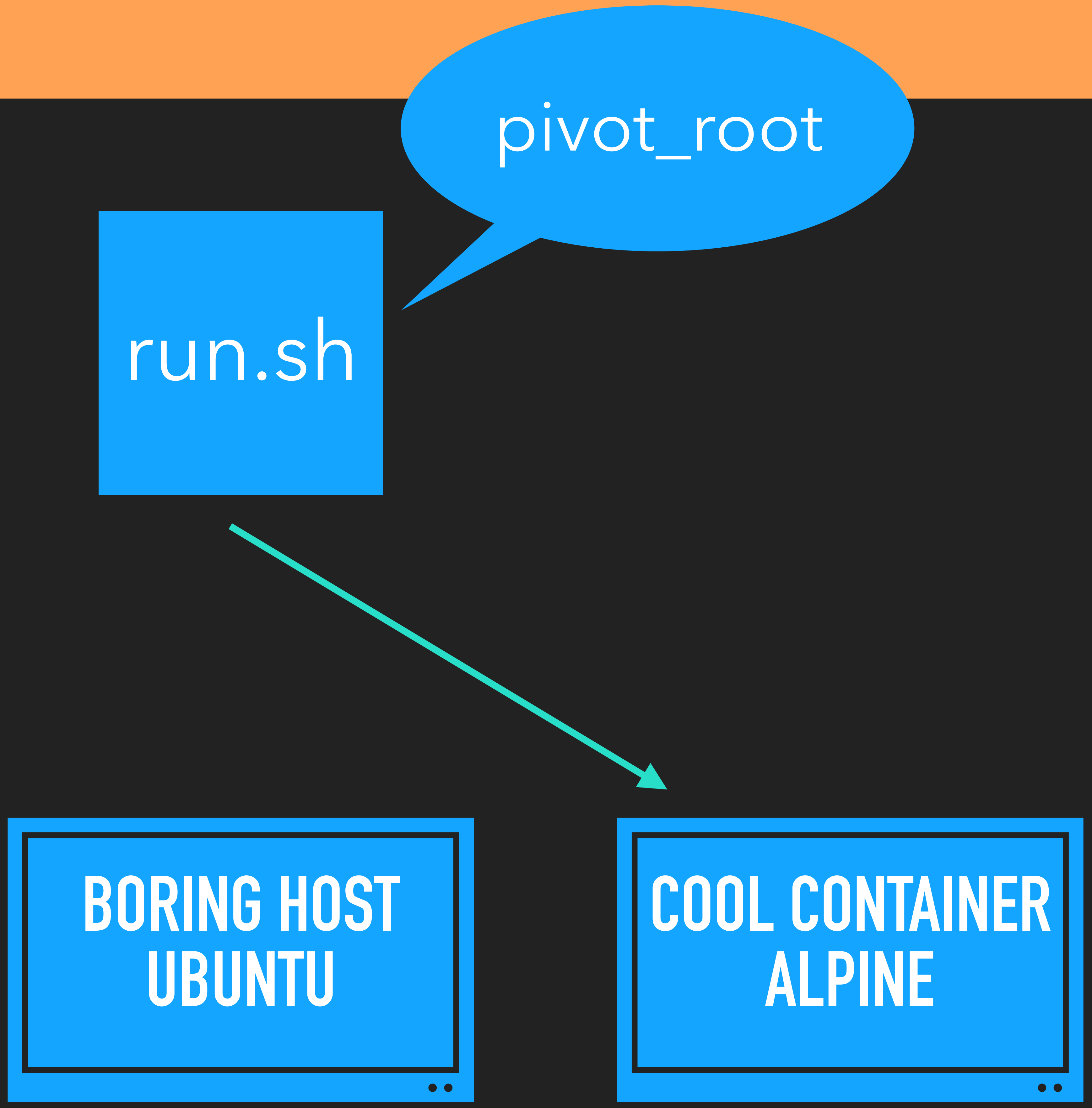
WHAT IS A CONTAINER?

ENCAPSULATION: PIVOT_ROOT



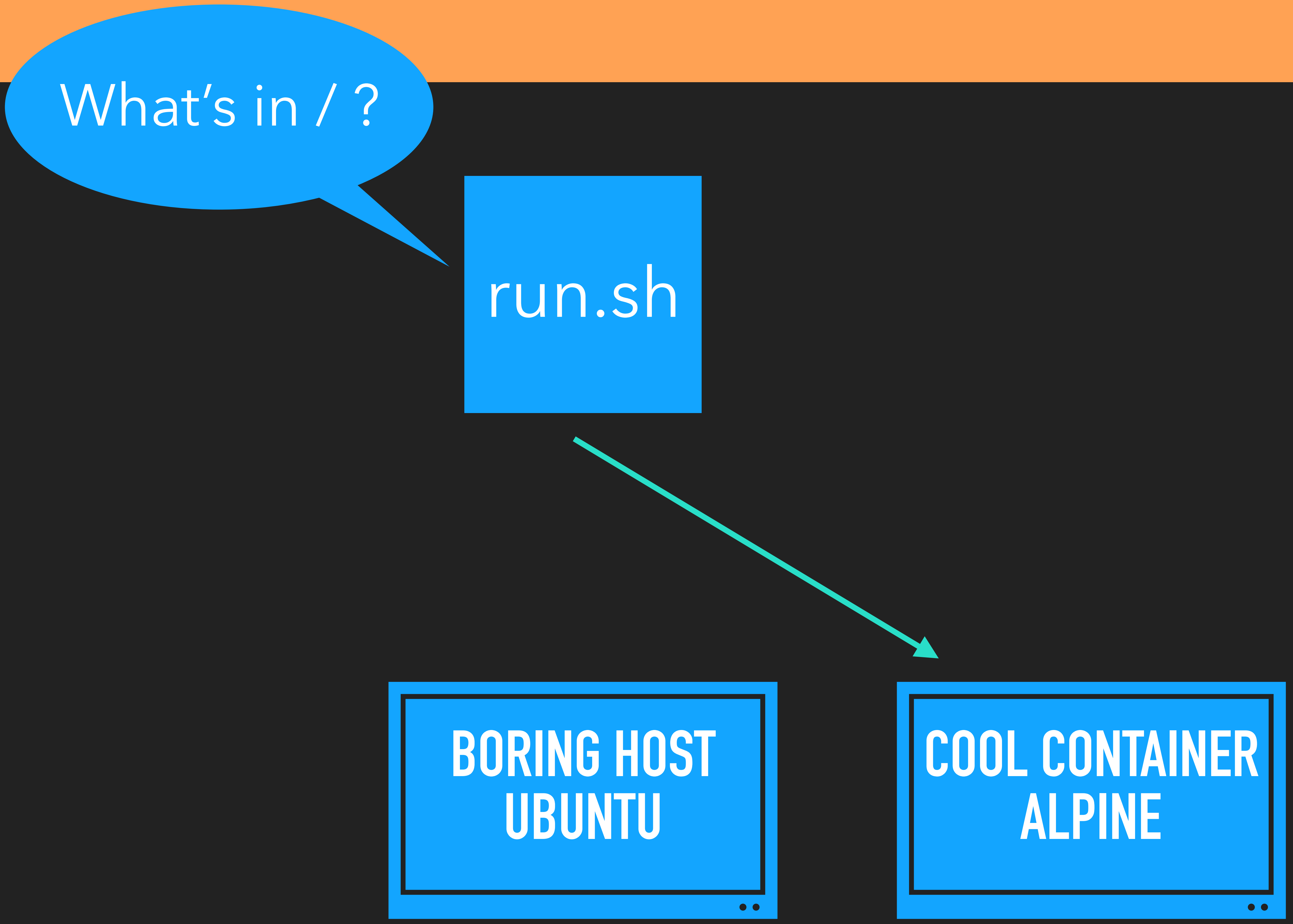
WHAT IS A CONTAINER?

ENCAPSULATION: PIVOT_ROOT



WHAT IS A CONTAINER?

ENCAPSULATION: PIVOT_ROOT



What's in / ?

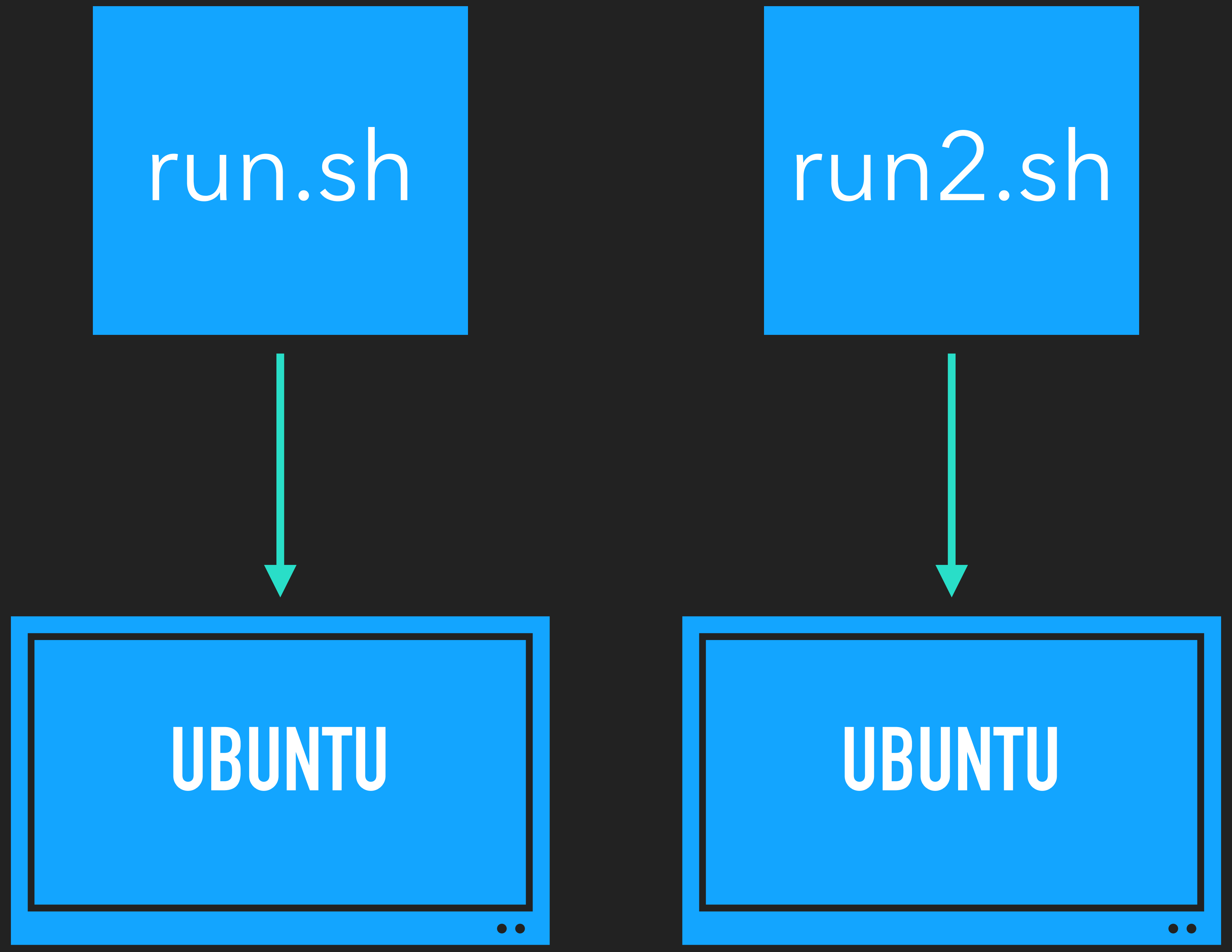
run.sh

BORING HOST
UBUNTU

COOL CONTAINER
ALPINE

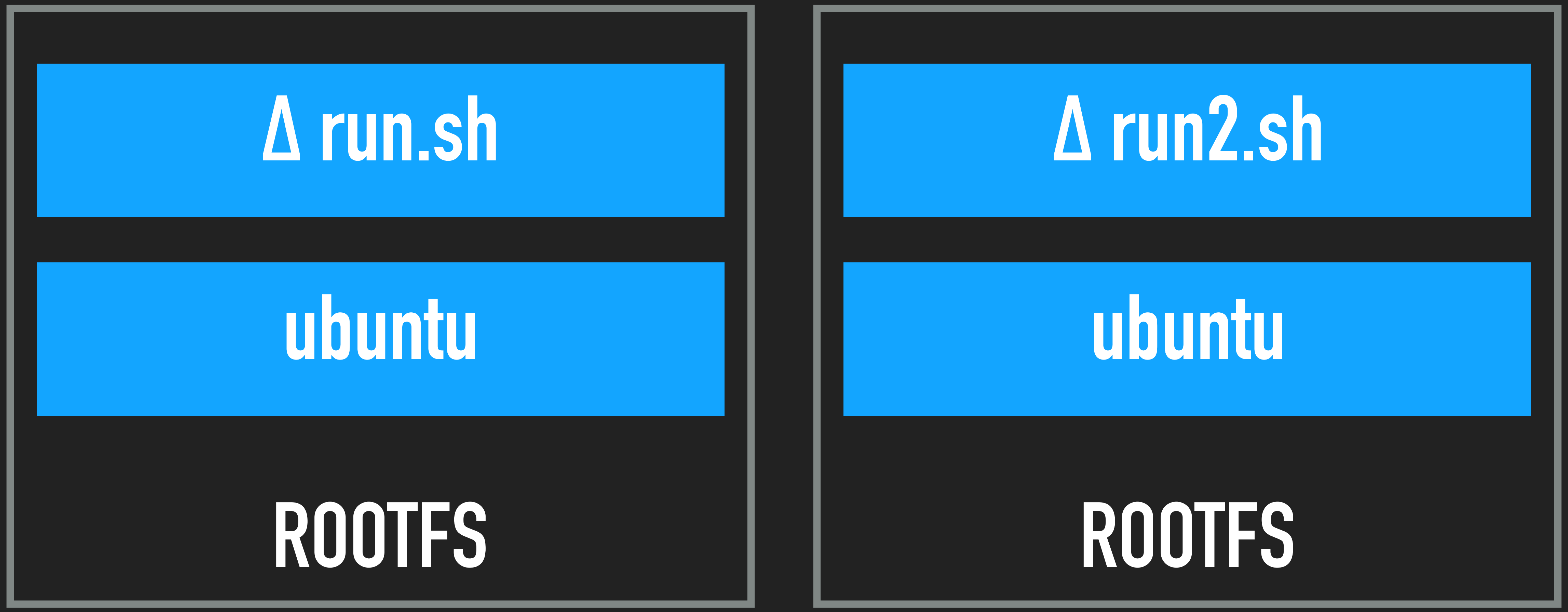
WHAT IS A CONTAINER?

ENCAPSULATION: LAYERED FS



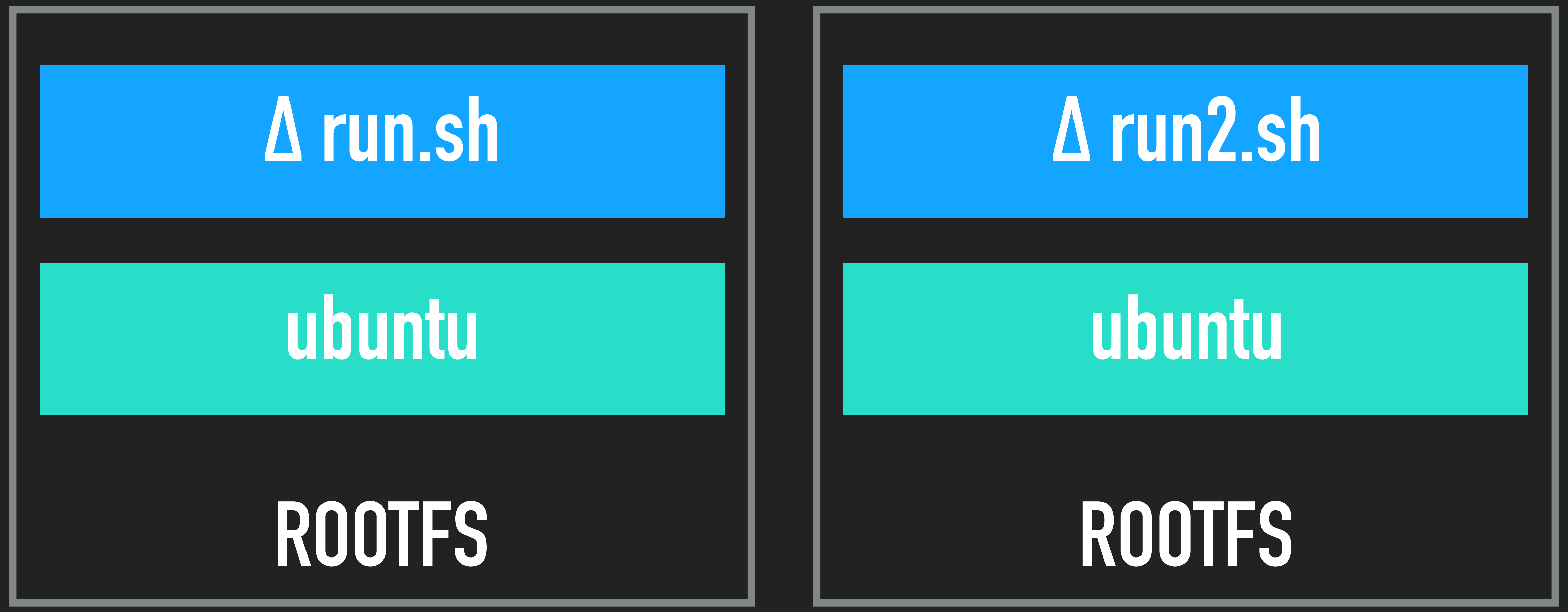
WHAT IS A CONTAINER?

ENCAPSULATION: LAYERED FS



WHAT IS A CONTAINER?

ENCAPSULATION: LAYERED FS



Δ run.sh

ubuntu

ROOTFS

Δ run2.sh

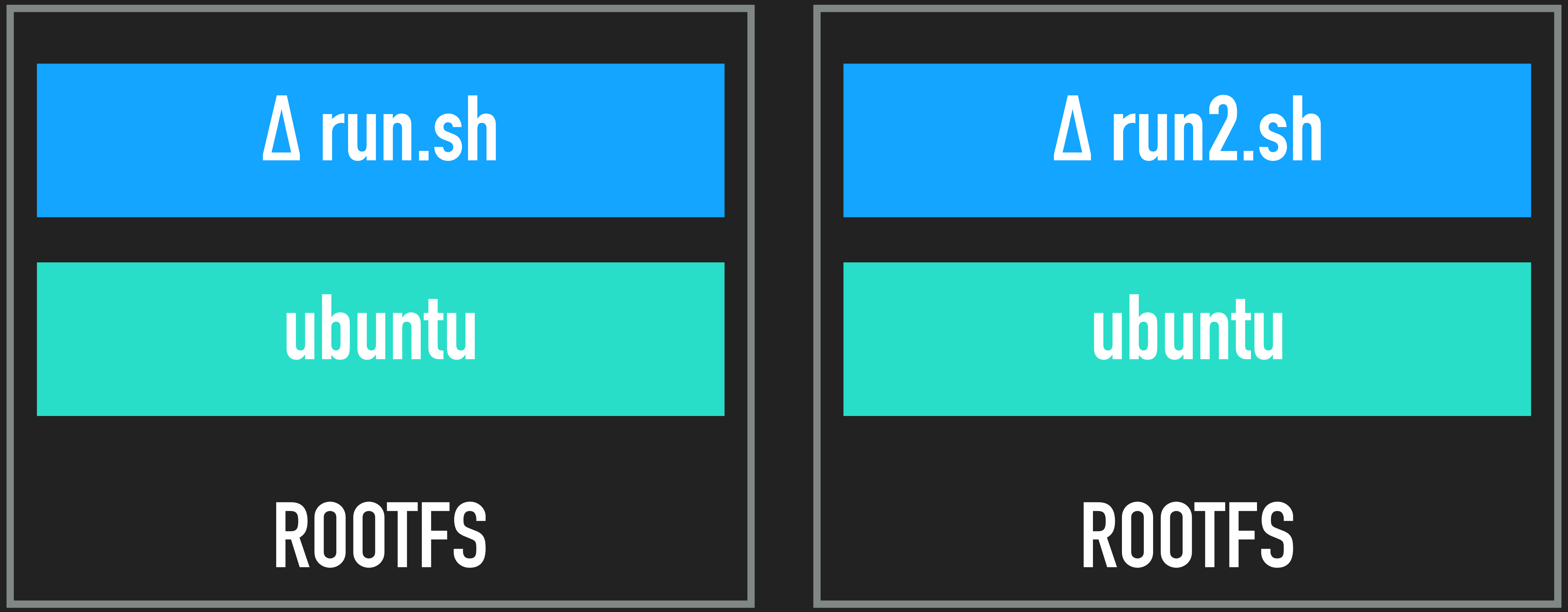
ubuntu

ROOTFS

CACHED LAYERS!

WHAT IS A CONTAINER?

ENCAPSULATION: LAYERED FS



Δ run.sh

ubuntu

ROOTFS

Δ run2.sh

ubuntu

ROOTFS

EFFICIENT SHIPPING!

WHAT IS A CONTAINER?

- ✓Isolation
- ✓Resource Sharing

- ✓Encapsulation

WHAT IS A CONTAINER?

STANDARDS FTW!

OCI

- Interoperable
- Standard standard shipping + runtime container format

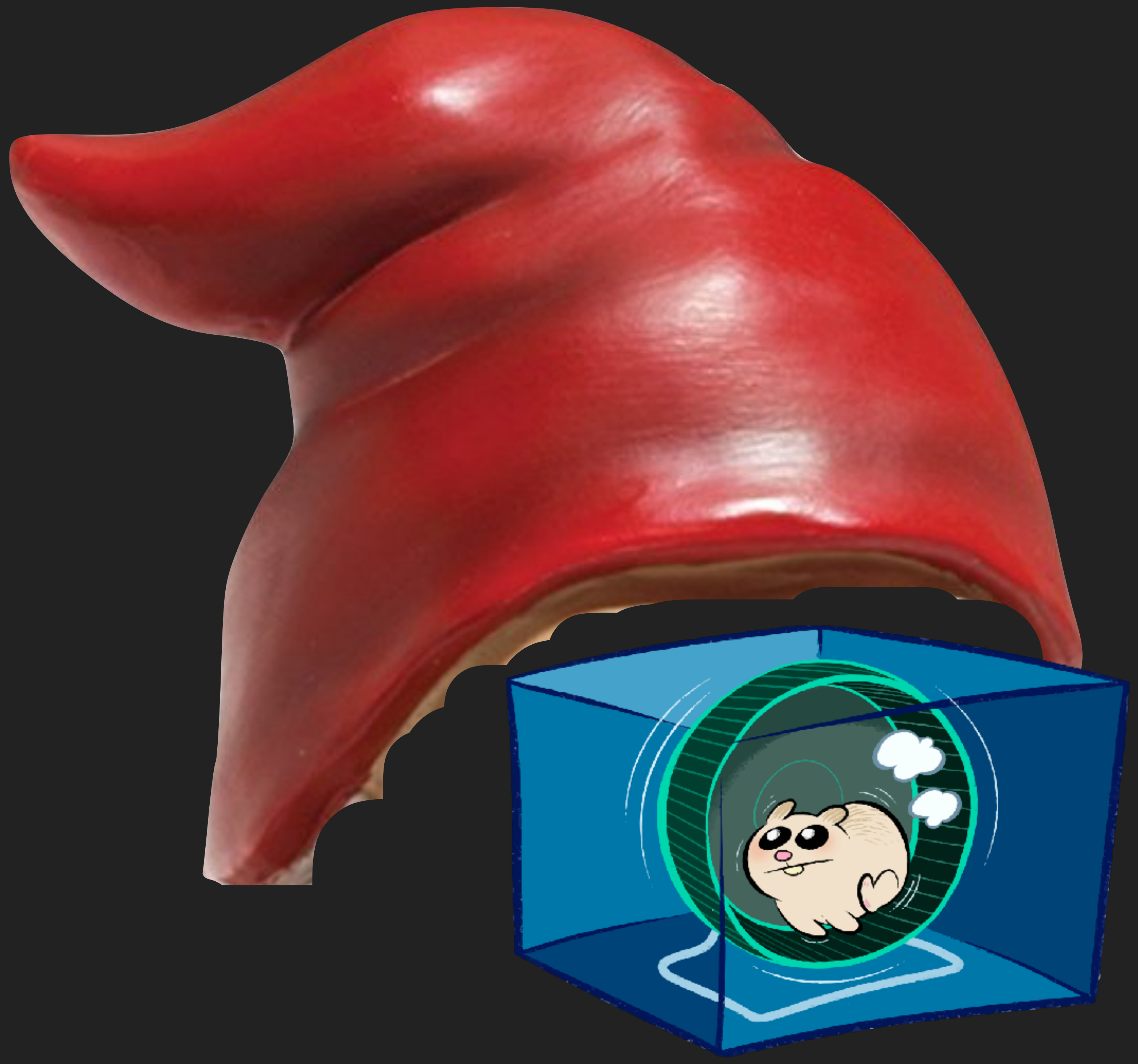
WHAT IS A CONTAINER?

STANDARDS FTW!

- Small, simple
- Standard
- Common low-level code (docker, k8s, cf..)

WHAT IS A CONTAINER?

STANDARDS FTW!



- Garden: CF Container Bindings
- Creates & Manages OCI Images/Bundles
- Runs ‘em with runC

CONTAINERS

- ✓ Isolation
- ✓ Resource Sha
- ✓ Encapsulation

CONTAINERS

- ✓ Isolation
- ✓ Resource Sha
- ✓ Encapsulation

SECURE

CONTAINERS

?

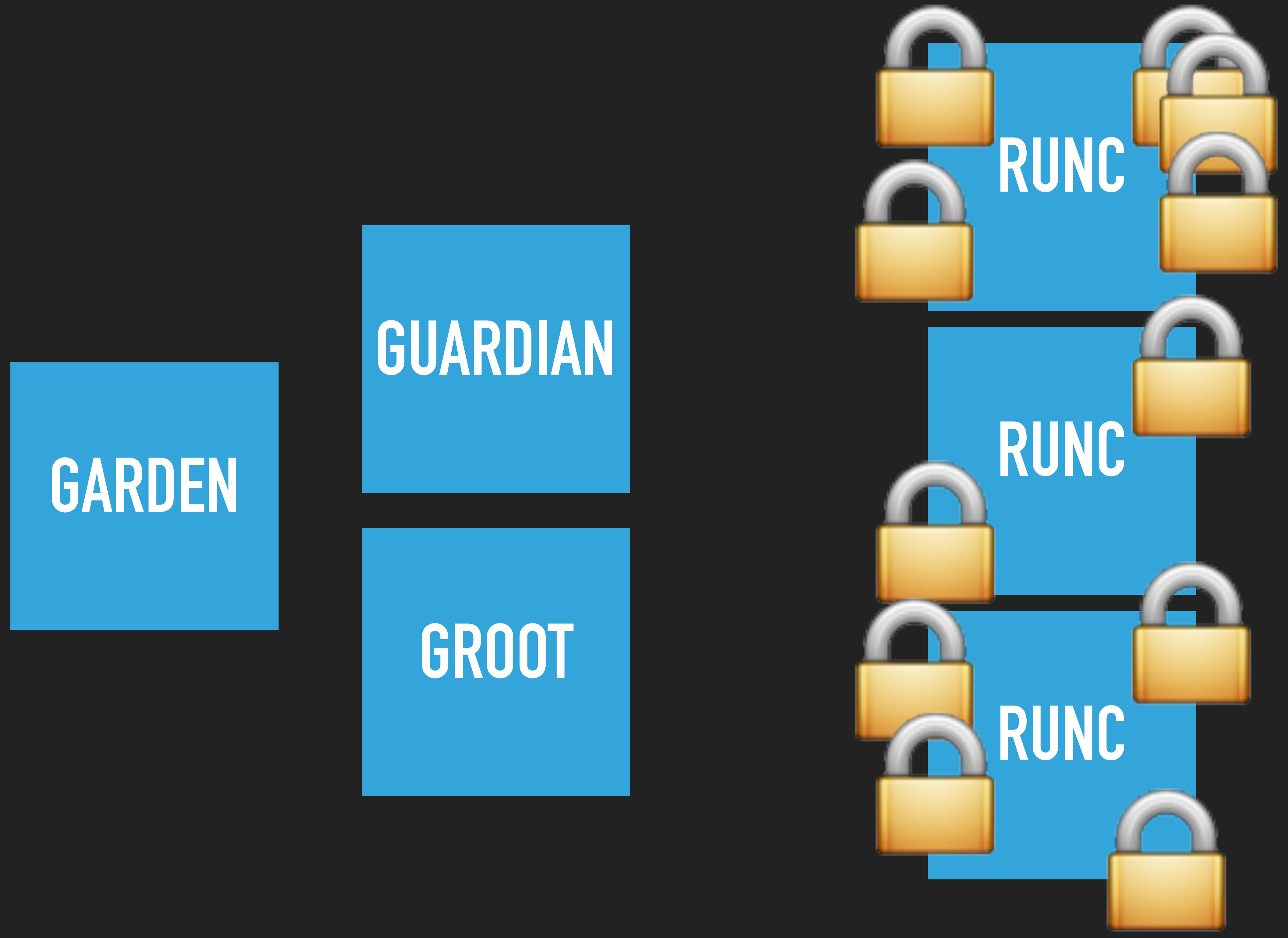
- ✓ Isolation
- ✓ Resource Sha
- ✓ Encapsulation

SECURE

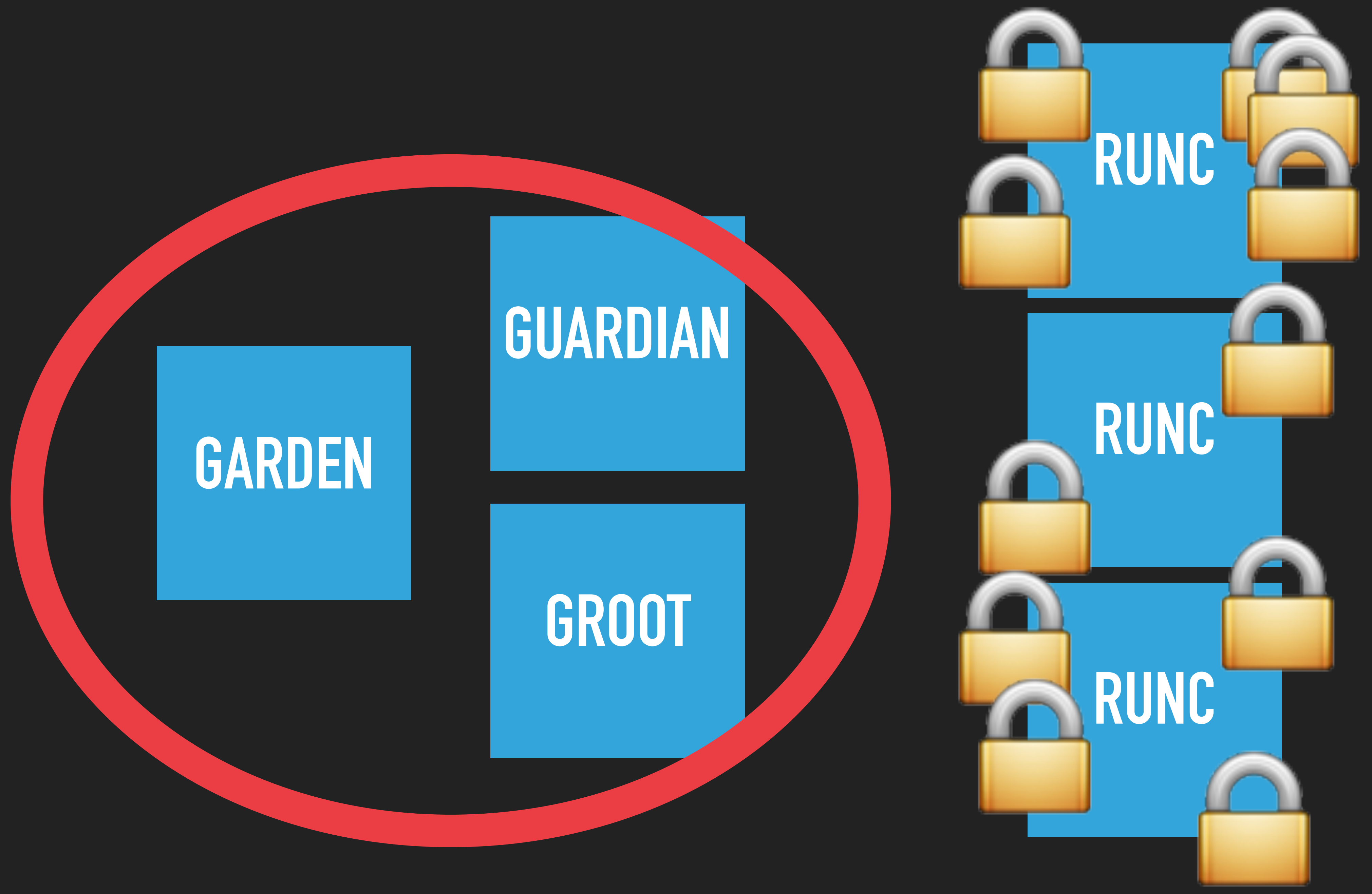
ARE WE SECURE?

ARE WE SECURE?

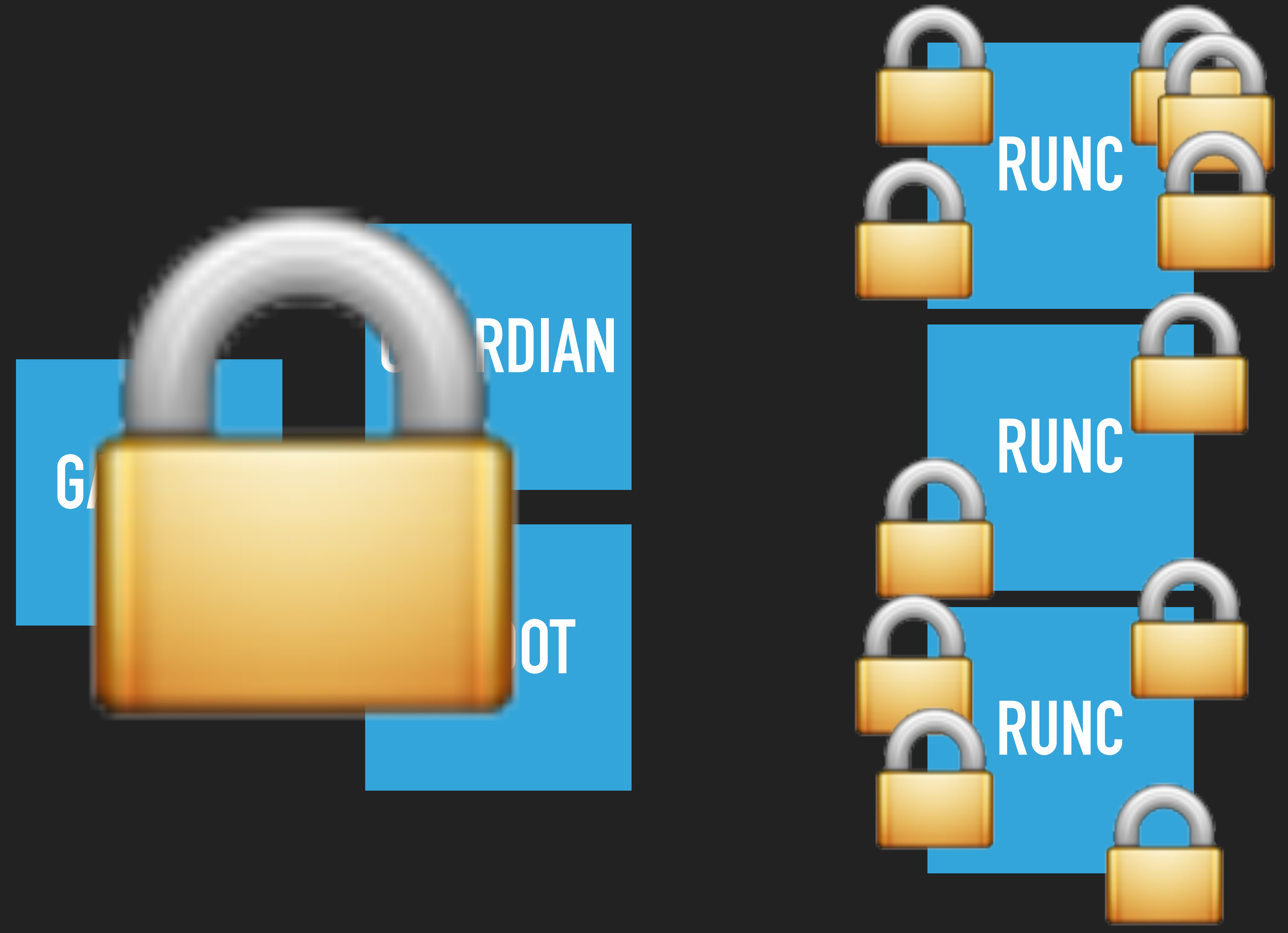
ARE WE SECURE?



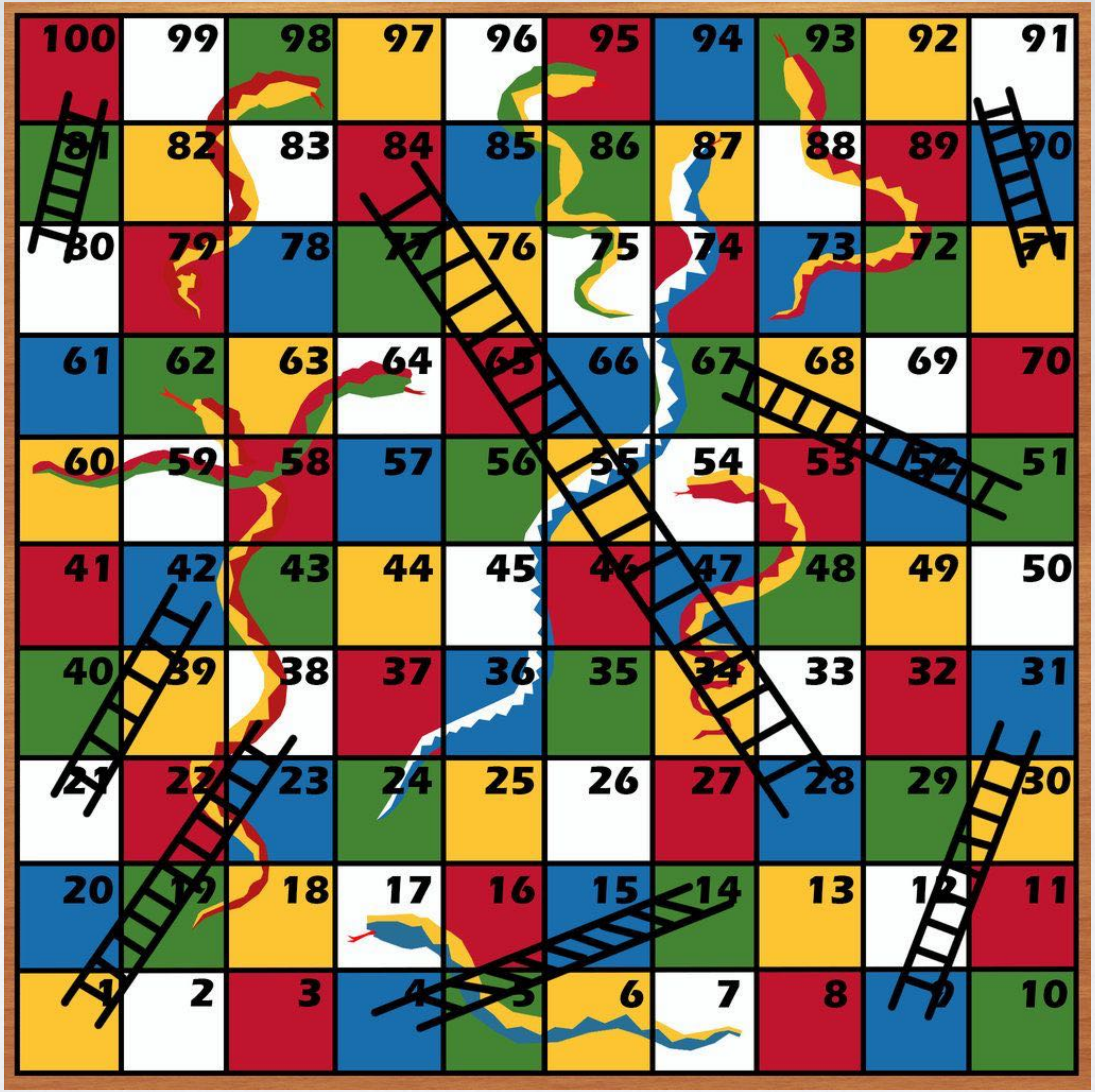
ARE WE SECURE?



ARE WE SECURE?



THE ROUTE TO ROOTLESS


100 99 98 97 96 95 94 93 92 91
81 82 83 84 85 86 87 88 89 90
80 79 78 77 76 75 74 73 72 71
61 62 63 64 65 66 67 68 69 70
60 59 58 57 56 55 54 53 52 51
41 42 43 44 45 46 47 48 49 50
40 39 38 37 36 35 34 33 32 31
21 22 23 24 25 26 27 28 29 30
20 19 18 17 16 15 14 13 12 11
1 2 3 4 5 6 7 8 9 10

MASSIVE PROPS & SHOUT OUTS!

- **Jessie Frazelle (@jessfraz)**
- **Aleksa Sarai (@lordcyphar)**
- **Akihiro Suda (@_AkihiroSuda_)**

- …and many more

THE ROUTE TO ROOTLESS

THE BIG TRICK: USER NAMESPACES

IN REALITY

- Average Frustrated User
- No special permissions

IN REALITY

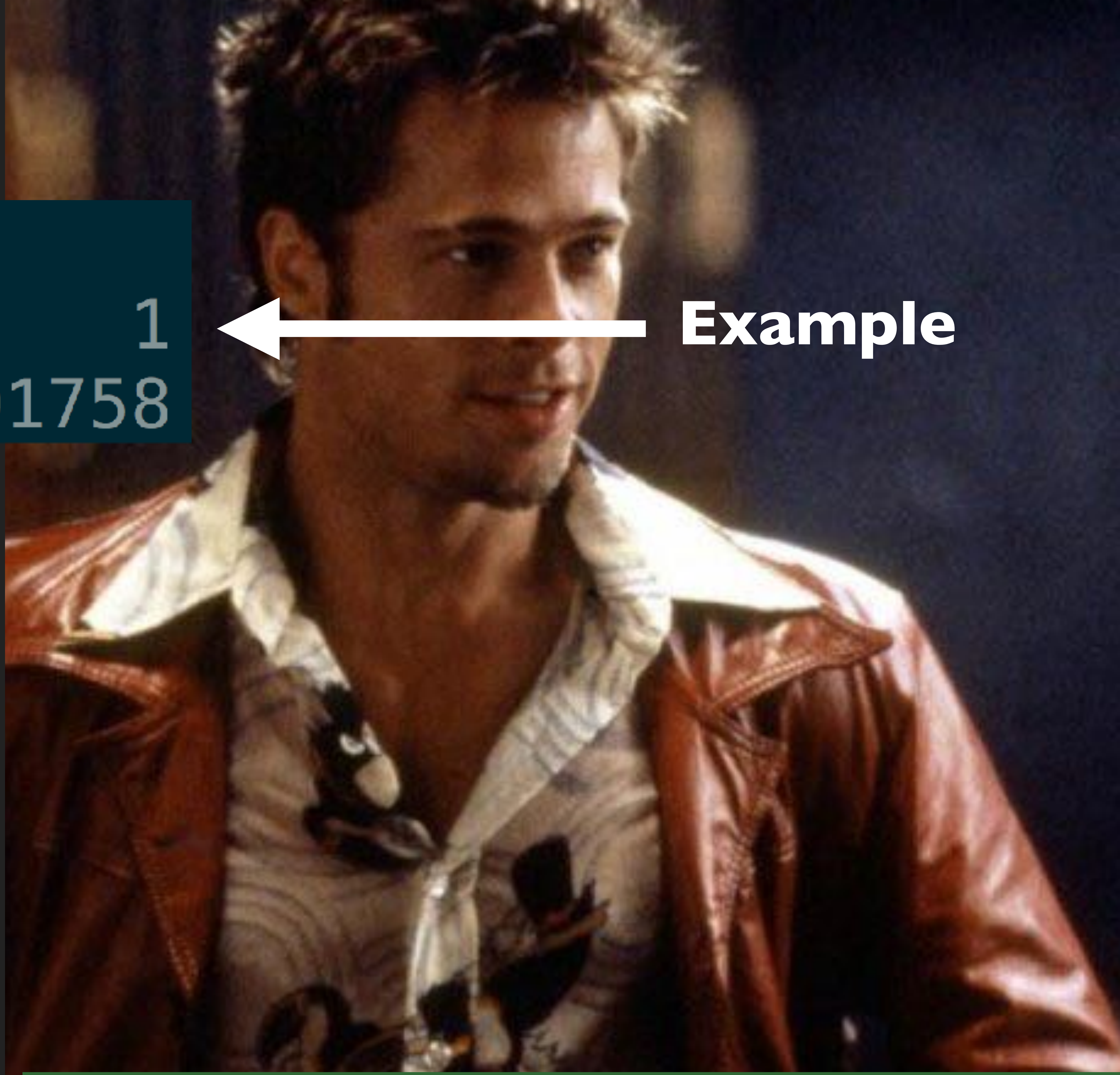
IN CONTAINER

- **I AM ROOT!**

IN CONTAINER

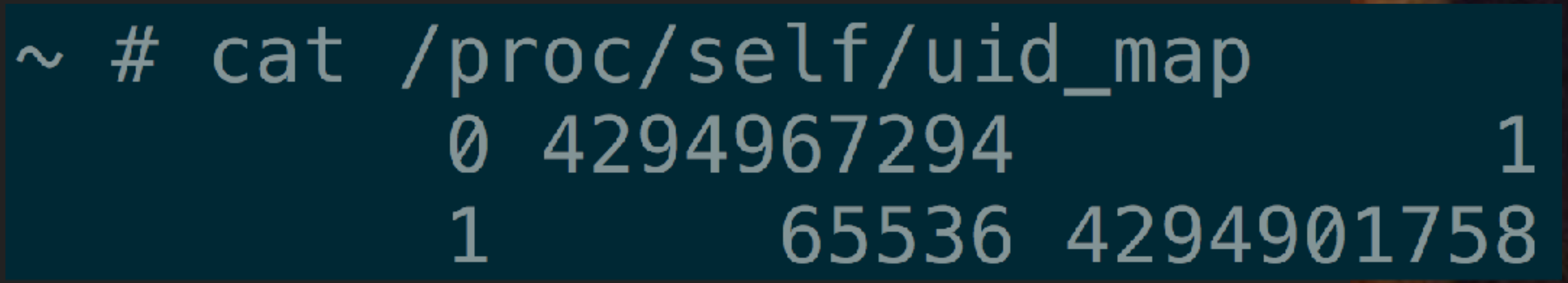
```
~ # cat /proc/self/uid_map
         0 4294967294          1
         1      65536 4294901758
```

- **I AM ROOT!**



Example

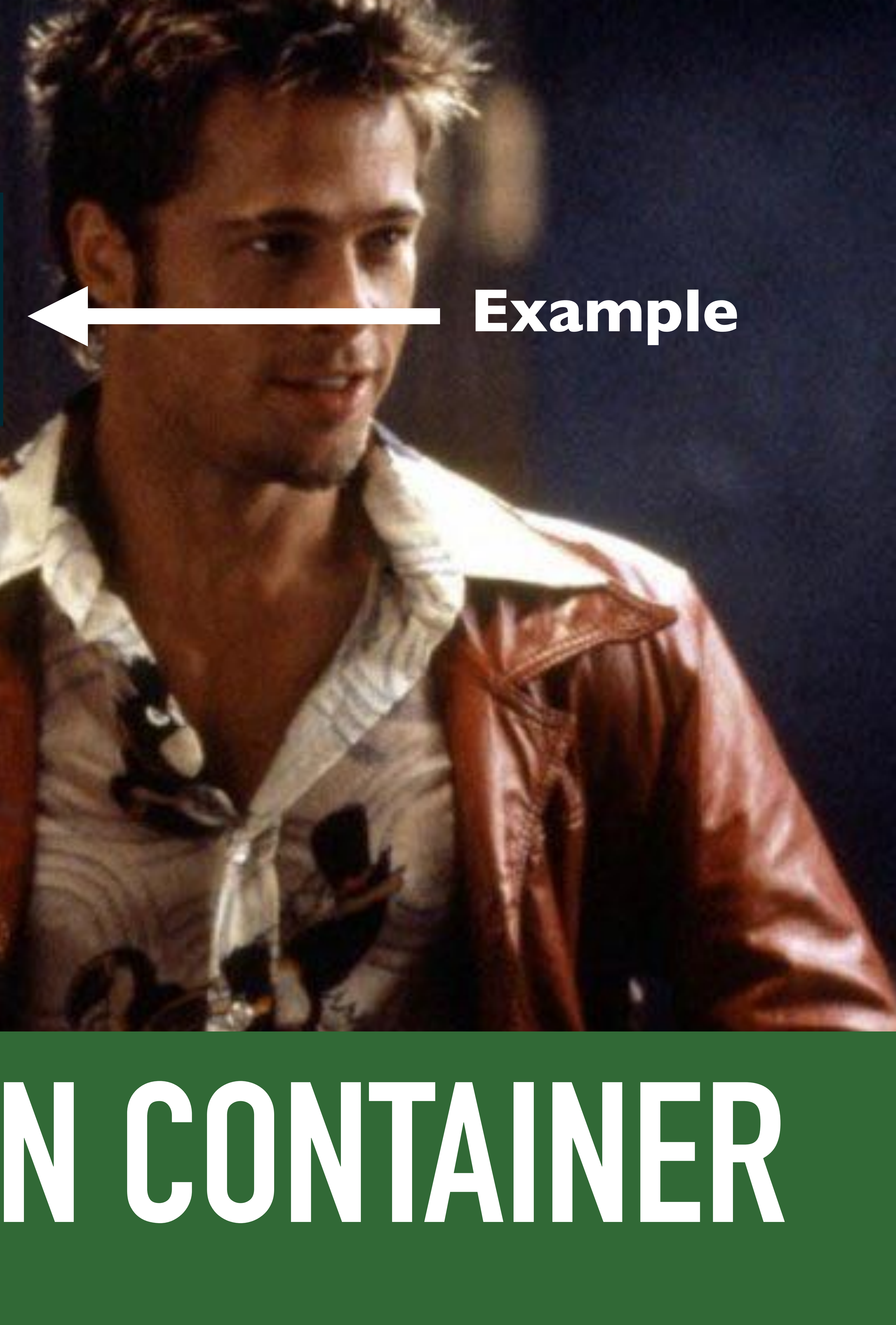
IN CONTAINER



```
~ # cat /proc/self/uid_map
         0 4294967294          1
         1      65536 4294901758
```

Example

- **I AM ROOT!**
- **(but only in this namespace and owned namespaces)**



IN CONTAINER

- Since Linux 3.8, *any* user can do this
- \o/

IN CONTAINER

- **CAP_SYS_ADMIN in user namespace lets you do Seccomp, AppArmor, other namespaces**
- **\o/ \o/**

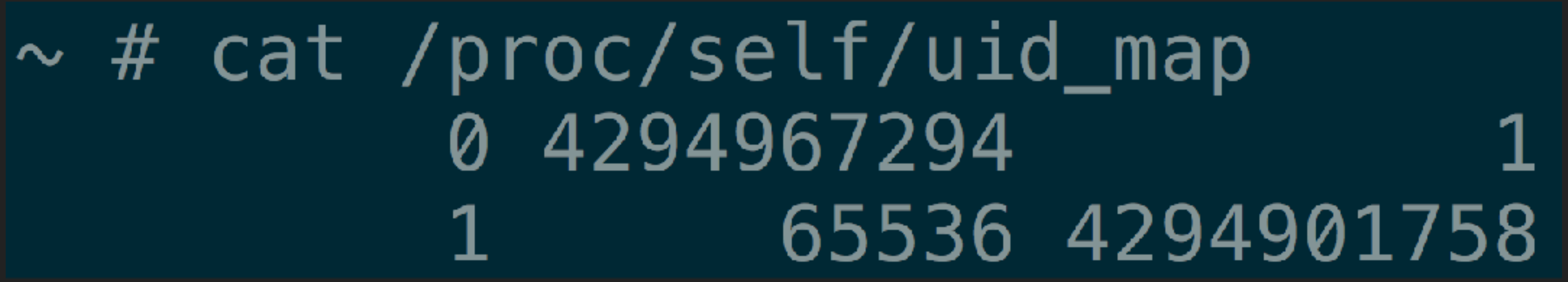
IN CONTAINER

THE BIG TRICK: USER NAMESPACES

- Any user can create a User Namespace
- You get to be root! (CAP_*)
- But only in that namespace, and namespaces created at the same time

THE ROUTE TO ROOTLESS

PROBLEM #1

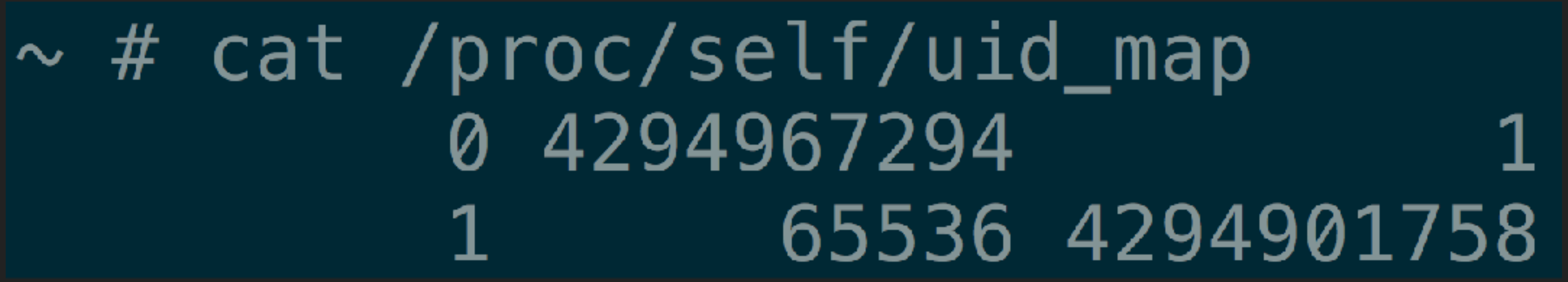


```
~ # cat /proc/self/uid_map
         0 4294967294          1
         1      65536 4294901758
```

- You only get 1 UID (your own)

THE ROUTE TO ROOTLESS

SOLUTION !



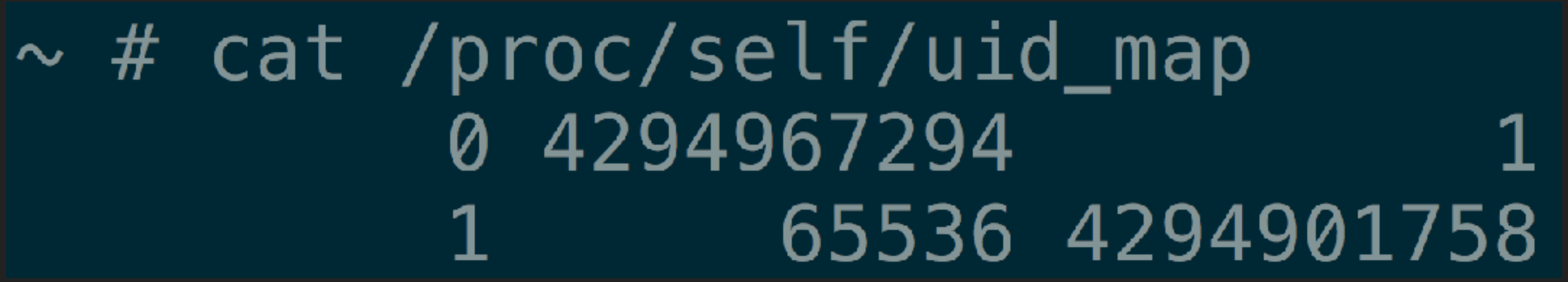
```
~ # cat /proc/self/uid_map
         0 4294967294          1
         1      65536 4294901758
```

- newuidmap
- /etc/subuid

THE ROUTE TO ROOTLESS

SOLUTION !

Cheating! But it's ok



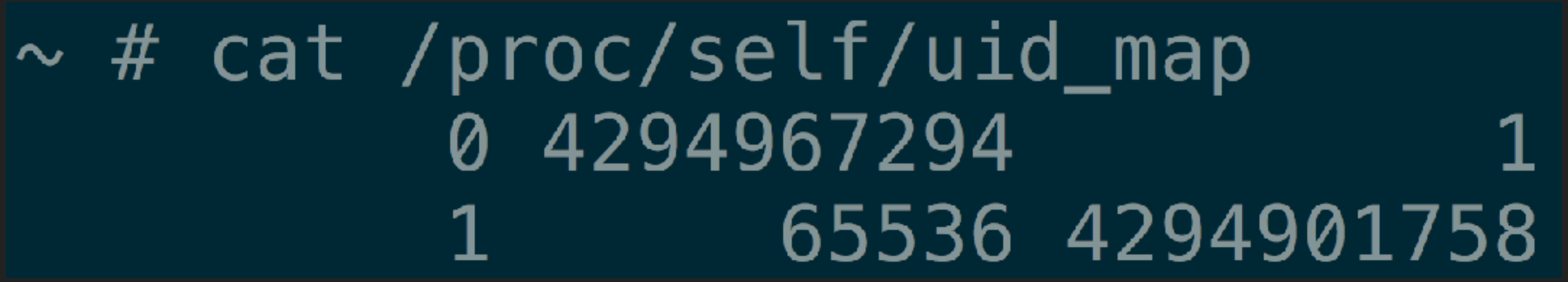
```
~ # cat /proc/self/uid_map
         0 4294967294          1
         1      65536 4294901758
```

- **newuidmap**
- **/etc/subuid**

THE ROUTE TO ROOTLESS

SOLUTION !

Cheating! But it's ok



```
~ # cat /proc/self/uid_map
         0 4294967294          1
         1      65536 4294901758
```

- **newuidmap**
- **/etc/subuid**
- **PRed runc \o/**

THE ROUTE TO ROOTLESS

Isolation

Resource Sharing

Encapsulation

THE ROUTE TO ROOTLESS

✓ Isolation

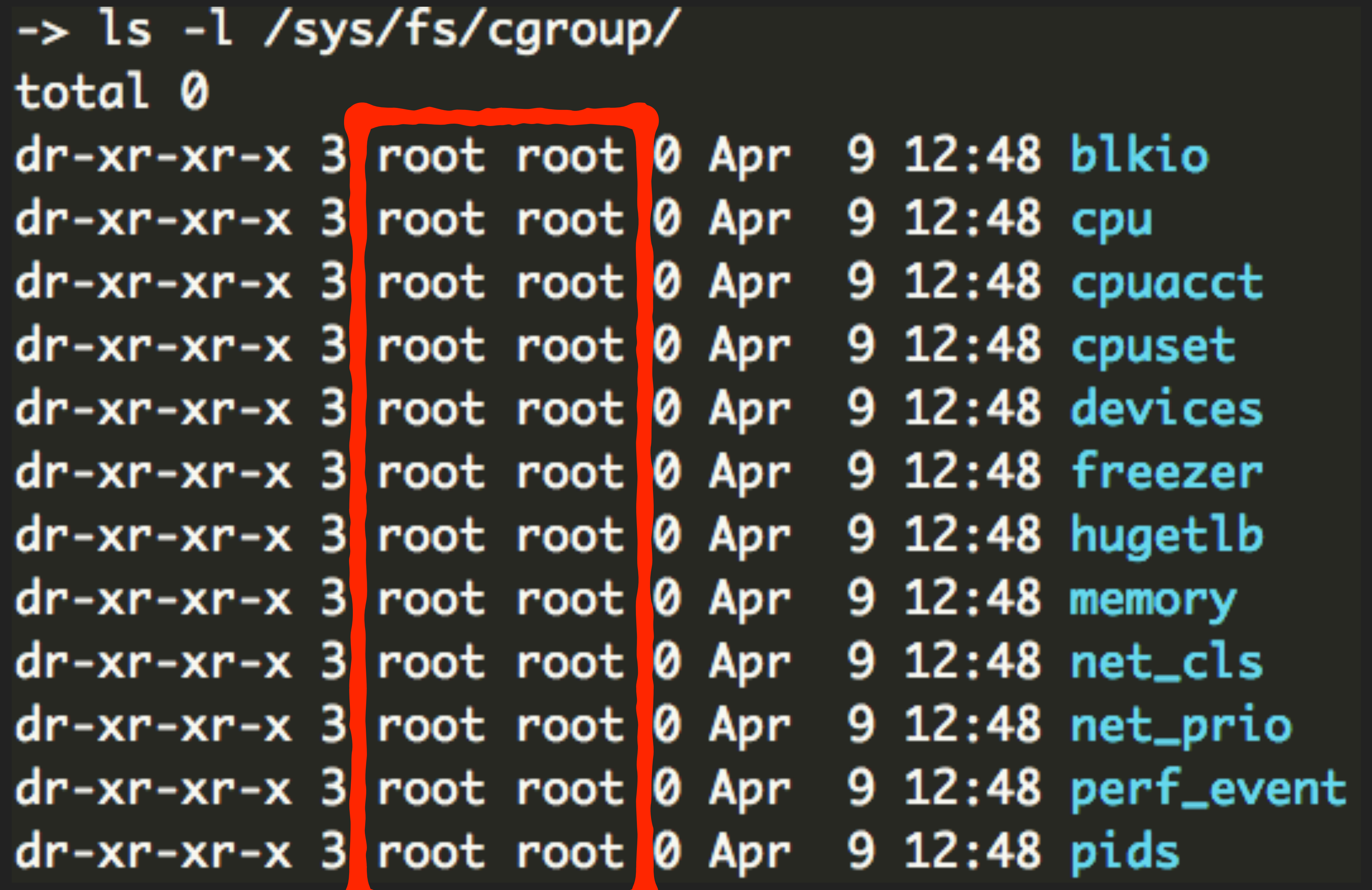
Resource Sharing

Encapsulation

THE ROUTE TO ROOTLESS

PROBLEM #2: CGROUPS :(

- Don’t work unprivileged (yet)

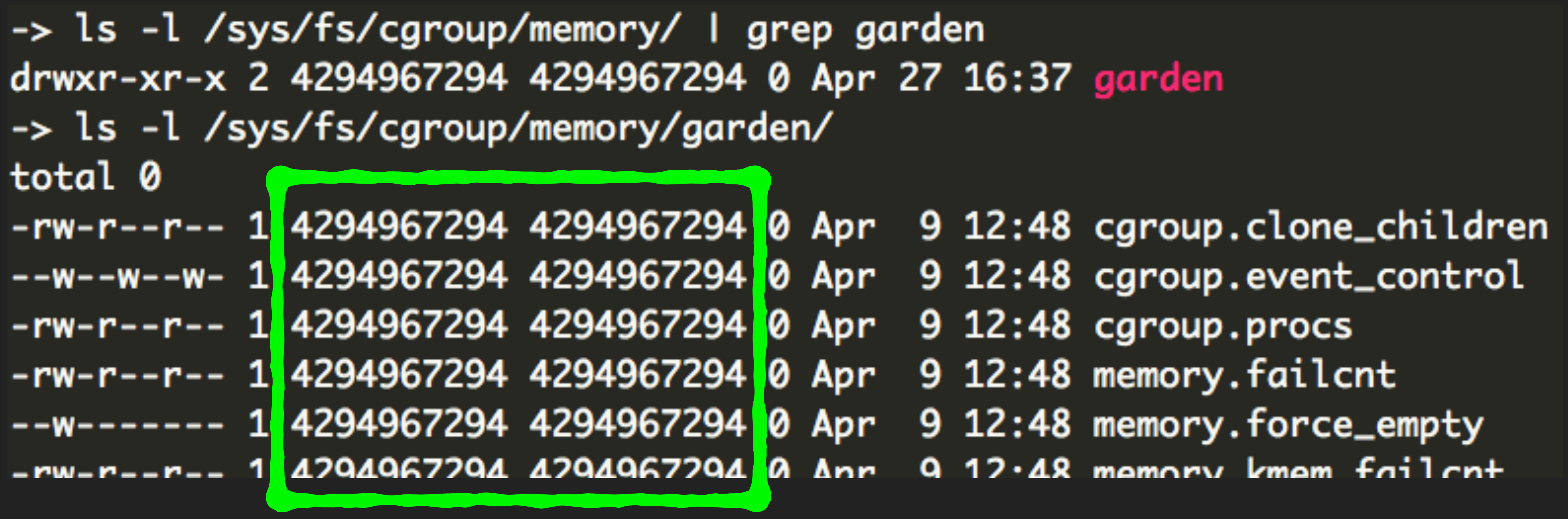


```
-> ls -l /sys/fs/cgroup/
total 0
dr-xr-xr-x 3 root root 0 Apr  9 12:48 blkio
dr-xr-xr-x 3 root root 0 Apr  9 12:48 cpu
dr-xr-xr-x 3 root root 0 Apr  9 12:48 cpuacct
dr-xr-xr-x 3 root root 0 Apr  9 12:48 cpuset
dr-xr-xr-x 3 root root 0 Apr  9 12:48 devices
dr-xr-xr-x 3 root root 0 Apr  9 12:48 freezer
dr-xr-xr-x 3 root root 0 Apr  9 12:48 hugetlb
dr-xr-xr-x 3 root root 0 Apr  9 12:48 memory
dr-xr-xr-x 3 root root 0 Apr  9 12:48 net_cls
dr-xr-xr-x 3 root root 0 Apr  9 12:48 net_prio
dr-xr-xr-x 3 root root 0 Apr  9 12:48 perf_event
dr-xr-xr-x 3 root root 0 Apr  9 12:48 pids
```

THE ROUTE TO ROOTLESS

SOLUTION!

- chown cgroups during a privileged setup phase!

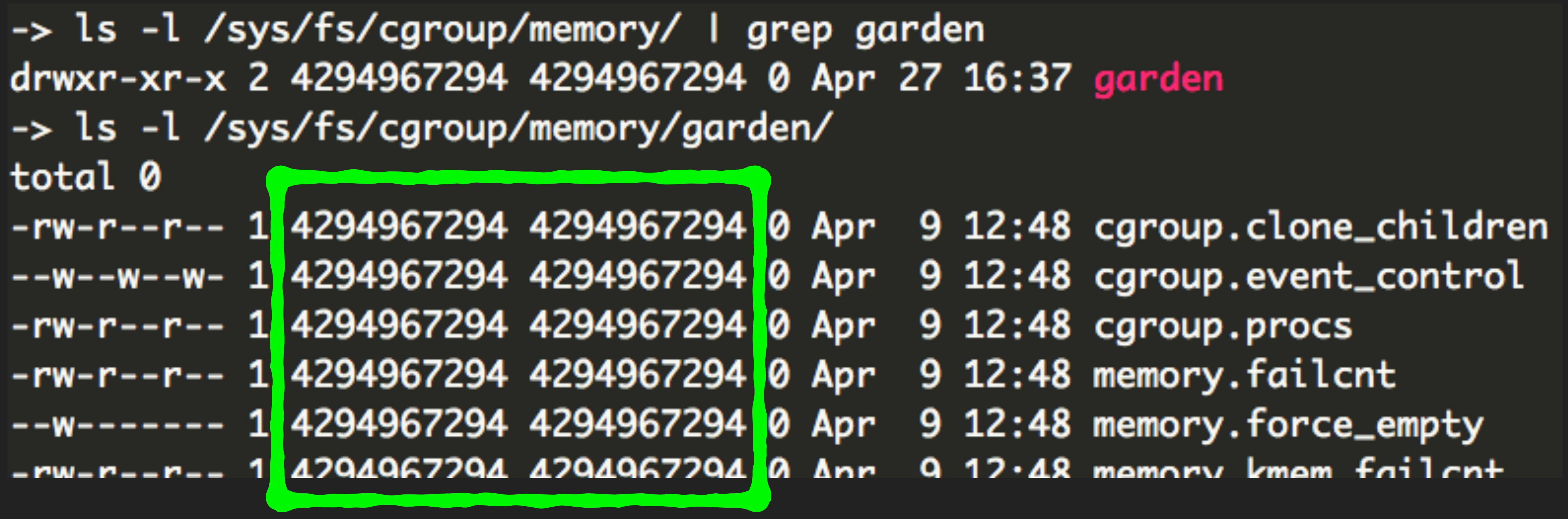


```
-> ls -l /sys/fs/cgroup/memory/ | grep garden
drwxr-xr-x 2 4294967294 4294967294 0 Apr 27 16:37 garden
-> ls -l /sys/fs/cgroup/memory/garden/
total 0
-rw-r--r-- 1 4294967294 4294967294 0 Apr  9 12:48 cgroup.clone_children
--w--w--w- 1 4294967294 4294967294 0 Apr  9 12:48 cgroup.event_control
-rw-r--r-- 1 4294967294 4294967294 0 Apr  9 12:48 cgroup.procs
-rw-r--r-- 1 4294967294 4294967294 0 Apr  9 12:48 memory.failcnt
--w------- 1 4294967294 4294967294 0 Apr  9 12:48 memory.force_empty
-rw-r--r-- 1 4294967294 4294967294 0 Apr  9 12:48 memory.kmem.failcnt
```

THE ROUTE TO ROOTLESS

SOLUTION!

- chown cgroups during a privileged setup phase!



```
-> ls -l /sys/fs/cgroup/memory/ | grep garden
drwxr-xr-x 2 4294967294 4294967294 0 Apr 27 16:37 garden
-> ls -l /sys/fs/cgroup/memory/garden/
total 0
-rw-r--r-- 1 4294967294 4294967294 0 Apr  9 12:48 cgroup.clone_children
--w--w--w- 1 4294967294 4294967294 0 Apr  9 12:48 cgroup.event_control
-rw-r--r-- 1 4294967294 4294967294 0 Apr  9 12:48 cgroup.procs
-rw-r--r-- 1 4294967294 4294967294 0 Apr  9 12:48 memory.failcnt
--w------- 1 4294967294 4294967294 0 Apr  9 12:48 memory.force_empty
-rw-r--r-- 1 4294967294 4294967294 0 Apr  9 12:48 memory.kmem.failcnt
```

- PRed runc \o/

THE ROUTE TO ROOTLESS

✓ Isolation

Resource Sharing

Encapsulation

THE ROUTE TO ROOTLESS

✓ Isolation

Resource Sharing

Encapsulation

THE ROUTE TO ROOTLESS

✓ ds

Ro

E

THE ROUTE TO ROOTLESS

PROBLEM #3: FILESYSTEMS

Mnt

(Namespace)

+

Pivot_Root

(Syscall)

Initial Namespace	“Container”
/path/to/mycontainer/rootfs	/
/path/to/mycontainer/rootfs/home/	/home

THE ROUTE TO ROOTLESS

PROBLEM #3: FILESYSTEMS

Allowed with CAP_SYS_ADMIN in User Namespace!

Mnt

(Namespace)

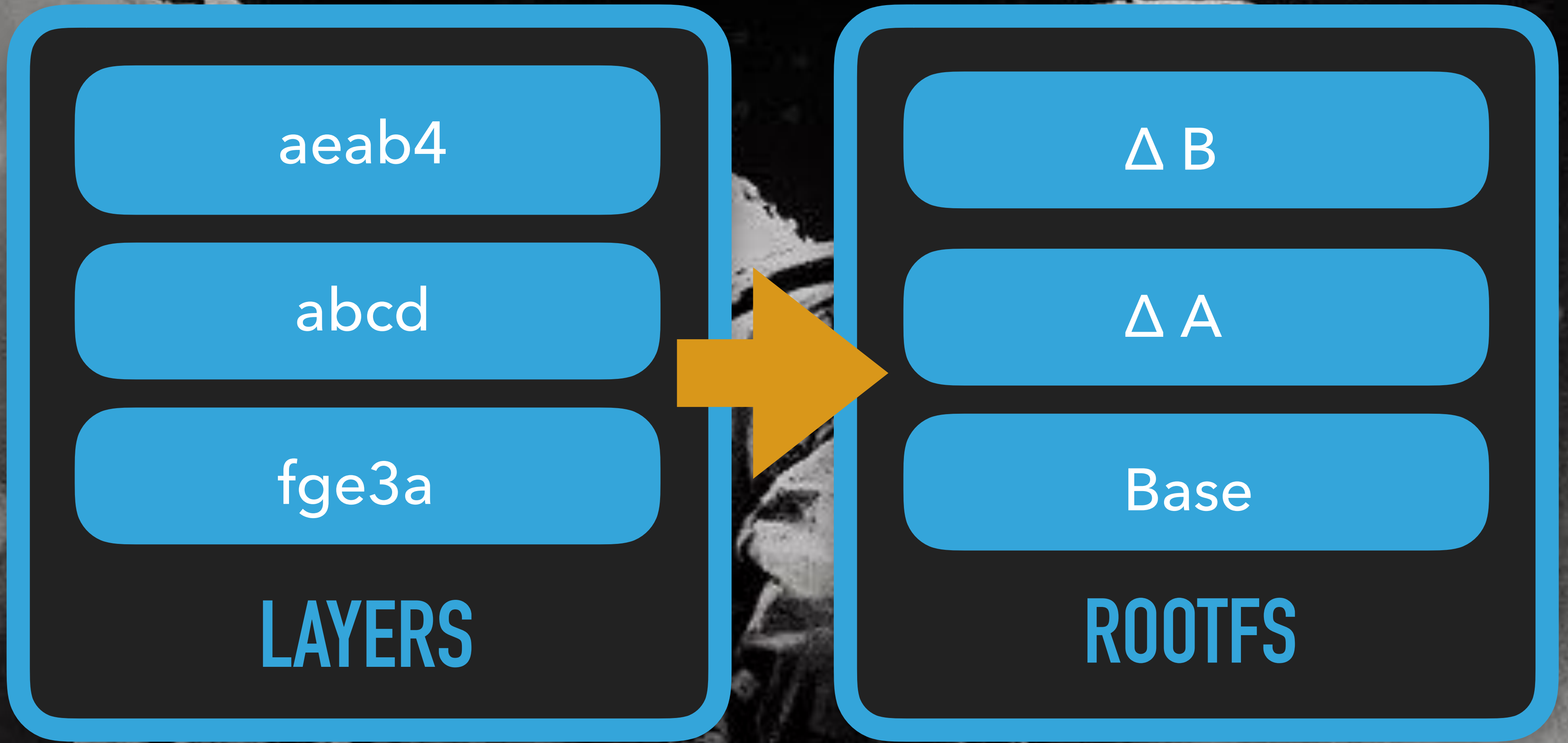
+

Pivot_Root

(Syscall)

THE ROUTE TO ROOTLESS

PROBLEM #3: FILESYSTEMS



aeab4

abcd

fge3a

LAYERS

Δ B

Δ A

Base

ROOTFS

- Layered Filesystems
- Copy-on-Write

THE ROUTE TO ROOTLESS

PROBLEM #3: FILESYSTEMS

copy-on-write filesystems

- AUFS
- BTRFS
- Overlayfs

THE ROUTE TO ROOTLESS

PROBLEM #3: FILESYSTEMS

1. AUFS

- Run in production for ages
- Not in mainline kernel
- No way to do without root :(

THE ROUTE TO ROOTLESS

PROBLEM #3: FILESYSTEMS

2. BTRFS

- Can create a “snapshot” without root!
- Bit of root at startup but that’s fine

PROBLEM #3: FILESYSTEMS

2. BTRFS

- Exploded at scale once quotas were turned on :-(

THE ROUTE TO ROOTLESS

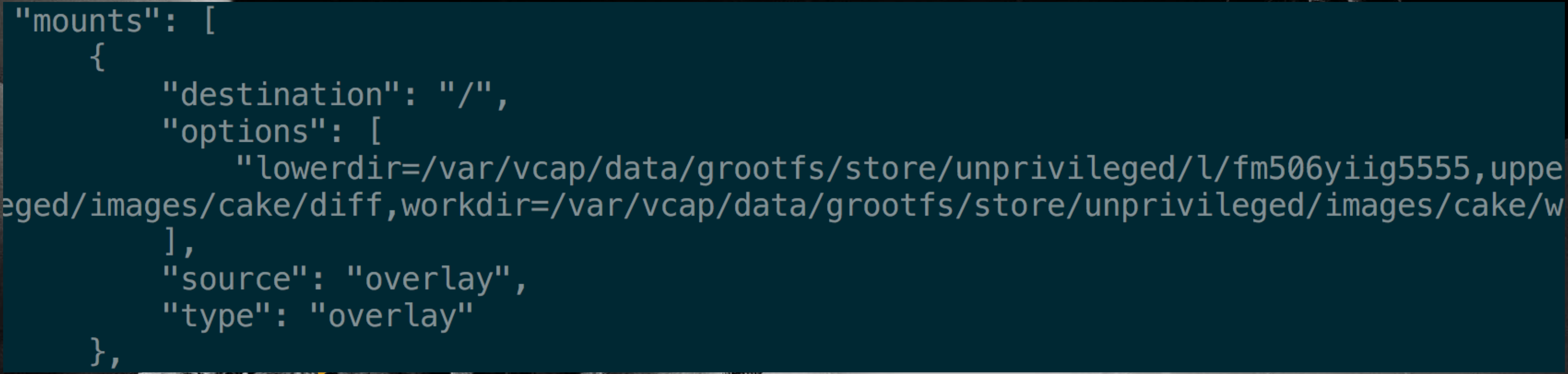
PROBLEM #3: FILESYSTEMS

3. OverlayFS

- Mainline kernel
- Allowed inside User Namespace on Ubuntu!

THE ROUTE TO ROOTLESS

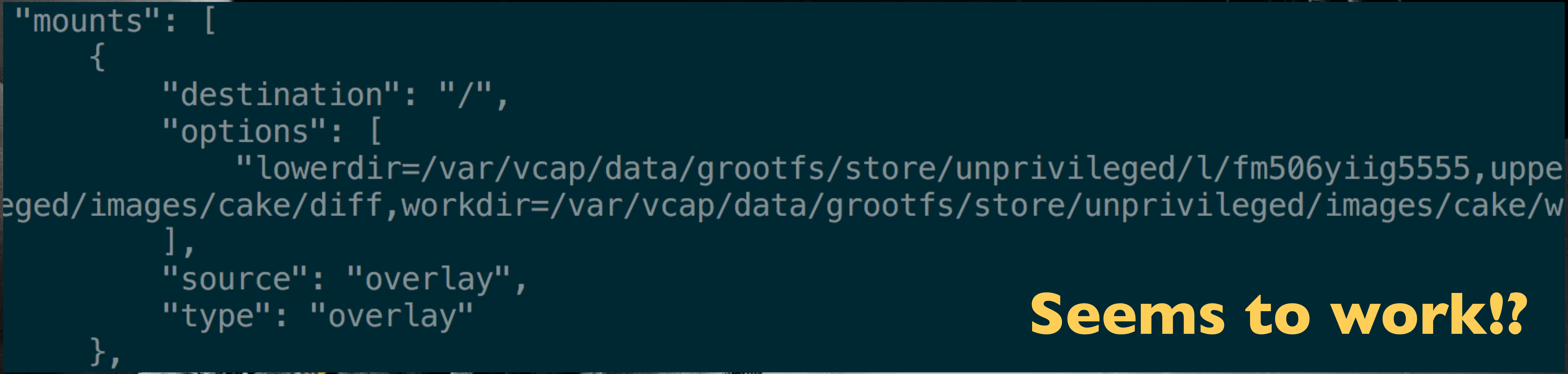
SOLUTION! OVERLAY IN USERNS



```
"mounts": [
    {
        "destination": "/",
        "options": [
            "lowerdir=/var/vcap/data/grootfs/store/unprivileged/l/fm506yiig5555,uppe
eged/images/cake/diff,workdir=/var/vcap/data/grootfs/store/unprivileged/images/cake/w
        ],
        "source": "overlay",
        "type": "overlay"
    },
```

THE ROUTE TO ROOTLESS

SOLUTION! OVERLAY IN USERNS



```
"mounts": [
    {
        "destination": "/",
        "options": [
            "lowerdir=/var/vcap/data/grootfs/store/unprivileged/l/fm506yiig5555,uppe
eged/images/cake/diff,workdir=/var/vcap/data/grootfs/store/unprivileged/images/cake/w
        ],
        "source": "overlay",
        "type": "overlay"
    },
```

Seems to work!?

THE ROUTE TO ROOTLESS

✓ Isolation

✓ Resource Sharing

✓ Encapsulation

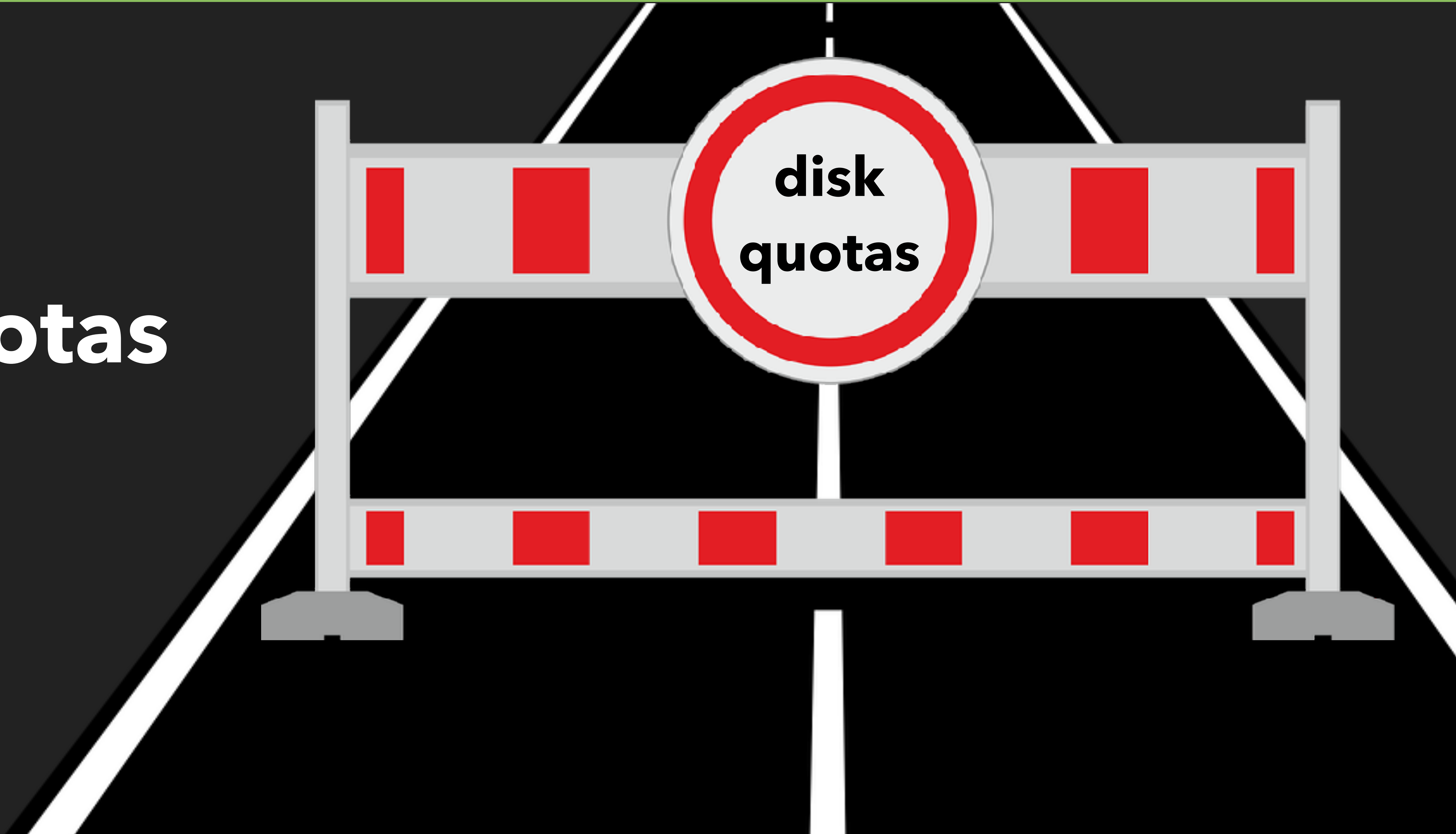
THE ROUTE TO ROOTLESS

REMAINING ROAD BLOCKS

REMAINING ROAD BLOCKS

DISK QUOTAS :(

- XFS for filesystem quotas
- Requires privilege



REMAINING ROAD BLOCKS

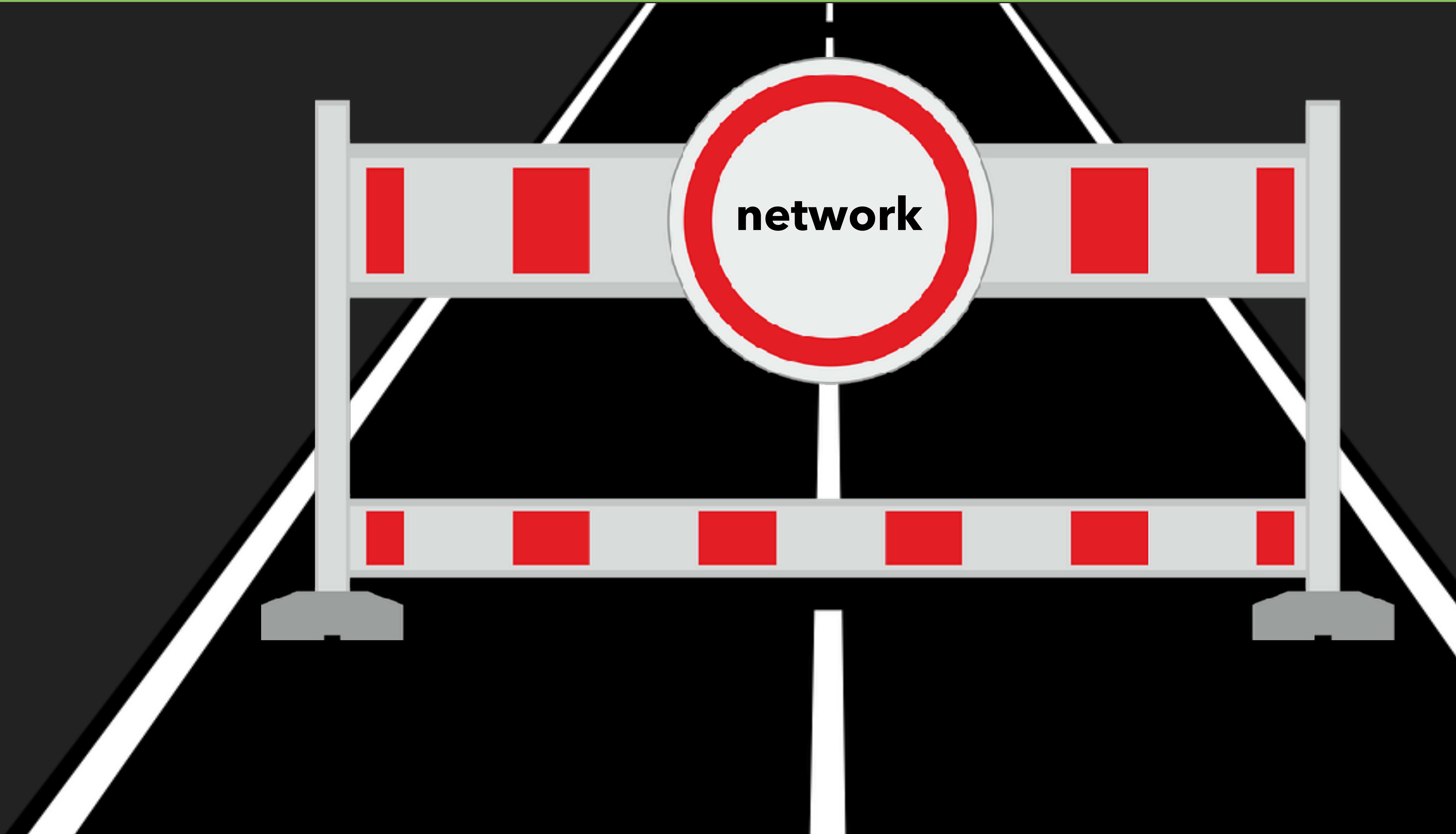
DISK QUOTAS :(

- Small, focused setuid binary

REMAINING ROAD BLOCKS

NETWORKING :(

- New net namespaces only have loopback
- Privileges required to configure others



REMAINING ROAD BLOCKS

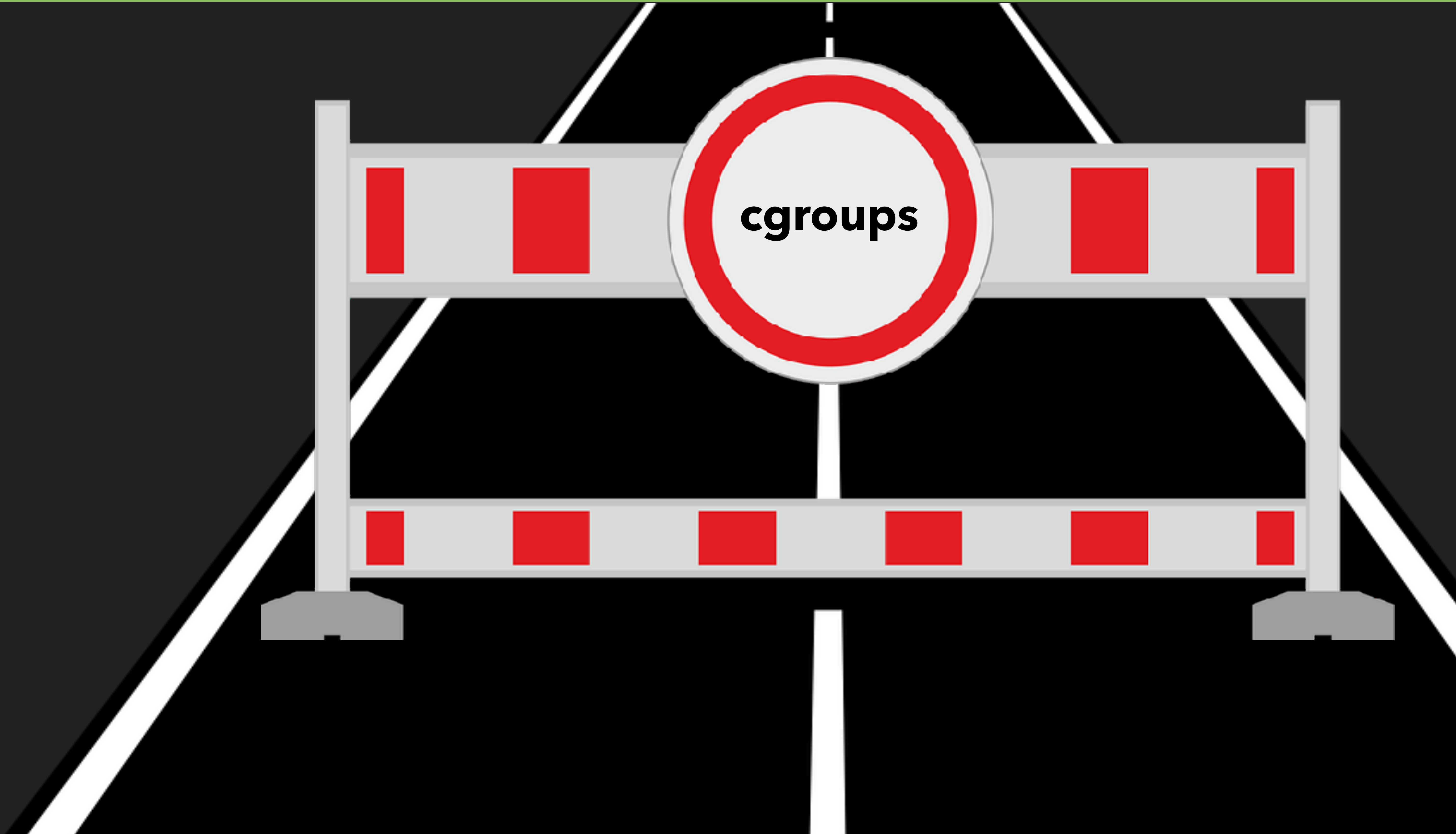
NETWORKING :(

- Some progress being made in this area
- Maybe rootless one day
- setuid binary for now

REMAINING ROAD BLOCKS

SETUP :(

- cgroup chowning



REMAINING ROAD BLOCKS

SETUP :(

- Setup runs before first container is created
- No user input
- Some ongoing effort to address this also

SUMMARY

1. WHY DO WE CARE?

2. CONTAINER SECURITY

3. ROOTLESS!

SUMMARY

DON'T WORRY BE HAPPY!

SUMMARY

PLAYING THE LONG GAME

- Reduce privilege where we can, when w
  - Some things take time, but proving th
- Break apart monoliths, to reduce privile
- Share technologies with the community
- [illegible]

SUMMARY

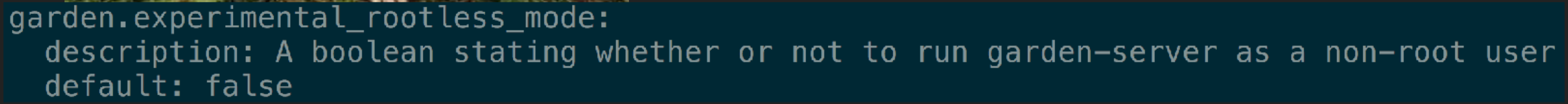
DOES IT WORK?!

- Hopefully!
- Passes all the CATS (Cloud Foundry Ac
- It's going out on PWS soon

SUMMARY

DOES IT WORK?!

- Hopefully!
- Passes all the CATS (Cloud Foundry Ac
- It's going out on PWS soon



```
garden.experimental_rootless_mode:
  description: A boolean stating whether or not to run garden-server as a non-root user
  default: false
```

@doctor_julz
julz.friedman@uk.ibm.com

@edking2
eking@pivotal.io

THANKS!