making a docker

containers from scratch

@tint:tint.red

goal

process isolation

a process has to look like it's on a separate machine without any sort of emulation

part 1: chroot

changes what folder the process sees the root directory as

```
mkdir container
cd container
doas chroot . /bin/sh
```

userspace

you need something to actually run in there

busybox provides a full posix userspace in a single binary

libc doesn't exist in the chroot so make sure to get the statically linked binary

distro	package	file
alpine	busybox-static	/bin/busybox.static
arch	extra/busybox	/bin/busybox
fedora	busybox	/sbin/busybox
ubuntu	busybox-static	/bin/busybox

```
mkdir -p bin
cp /bin/busybox.static bin/busybox
for i in sh ash ls cp mv wget ip ps echo mount ping; do
    ln -s busybox bin/"$i"
done
```

userspace but better

maybe a more complete environment is better. busybox doesn't provide things like TLS certificates

```
wget https://dl-cdn.alpinelinux.org/alpine/v3.19/releases/x86_64/alpine-
minirootfs-3.19.0-x86_64.tar.gz
tar xzvfp alpine-minirootfs-3.19.0-x86_64.tar.gz
doas chroot . /bin/sh
```

mounts

ps, dns, and others don't work, let's fix that

source: archwiki chroot article

```
doas mount -t proc /proc proc/
doas mount -o bind /sys sys/
doas mount -o bind /dev dev/
cp /etc/resolv.conf etc/resolv.conf
doas chroot . /bin/sh
```

problems

the chroot can still see:

- processes outside the chroot
- the network devices of the host
- the host's hostname
- the host's devices
- and more

teardown

```
doas umount proc
doas umount sys
doas umount dev
```

part 2: namespaces

when a process looks up a resource from the kernel, the kernel gives it a different view of the resources depending on its namespace

unshare

distro	package
alpine	util-linux-misc
arch	core/util-linux
fedora	util-linux-core
ubuntu	util-linux

```
doas unshare --ipc --mount --net --pid --uts --cgroup --time --mount-proc --root=.
--fork
```

mounts in namespaces

unshare mounts `/proc` for you, but `/sys` and `/dev` still need to exist

```
mount -t sysfs sys /sys
mount -t tmpfs dev /dev
mknod /dev/null c 1 3
mknod /dev/random c 1 8
mknod /dev/urandom c 1 9
mknod /dev/zero c 1 5
ln -s /proc/self/fd/0 /dev/stdin
ln -s /proc/self/fd/1 /dev/stdout
ln -s /proc/self/fd/2 /dev/stderr
ln -s /proc/self/fd /dev/fd
```

networking

the container only has `lo`. time to fix that

make sure the `iproute2` package is installed - busybox's `ip` doesn't support namespaces

on the host (run as root):

```
ip link add name vethhost type veth peer name vethcontainer netns /proc/$(pidof
unshare)/ns/net
ip addr add 10.255.0.1/24 dev vethhost
ip addr add fd00::1/64 dev vethhost
sysctl net.ipv4.ip_forward=1
sysctl net.ipv6.conf.all.forwarding=1
iptables -A FORWARD -i vethhost -j ACCEPT
iptables -A FORWARD -o vethhost -j ACCEPT
iptables -t nat -A POSTROUTING -s 10.255.0.0/24 -j MASQUERADE
ip6tables -A FORWARD -i vethhost -j ACCEPT
ip6tables -A FORWARD -o vethhost -j ACCEPT
ip6tables -t nat -A POSTROUTING -s fd00::/64 -j MASQUERADE
```

networking 2

in the container:

```
ip link set up vethcontainer
ip addr add 10.255.0.2/24 dev vethcontainer
ip addr add fd00::2/64 dev vethcontainer
ip route add default via 10.255.0.1
ip -6 route add default via fd00::1
```

volume mounts

just bind mount a folder before running `unshare`

```
mkdir host-home
doas mount -o bind /home host-home
```

teardown

the only things that don't remove themselves when you exit the container are the iptables commands and the volume mounts

```
doas iptables -D FORWARD -i vethhost -j ACCEPT
doas iptables -D FORWARD -o vethhost -j ACCEPT
doas iptables -t nat -D POSTROUTING -s 10.255.0.0/24 -j MASQUERADE
doas ip6tables -D FORWARD -i vethhost -j ACCEPT
doas ip6tables -D FORWARD -o vethhost -j ACCEPT
doas ip6tables -t nat -D POSTROUTING -s fd00::/64 -j MASQUERADE
doas umount host-home
cd ..
doas rm -rf container
```

automating all of this

```
docker run --rm -it alpine
```