Linux → kernel

grub → boot loader

systemd → initialization framework

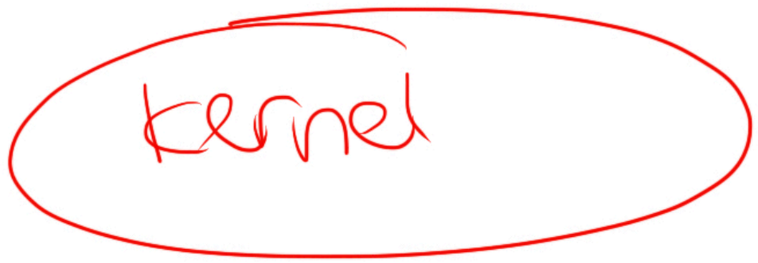
software

kernel

hardware

boot loader

initialization framework



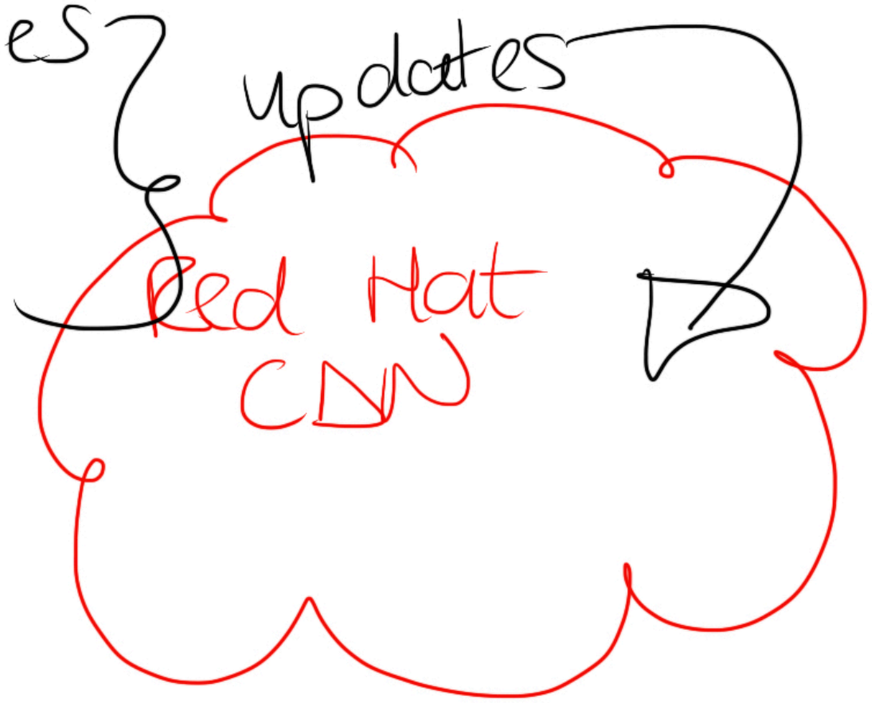
boot loader

initialization framework

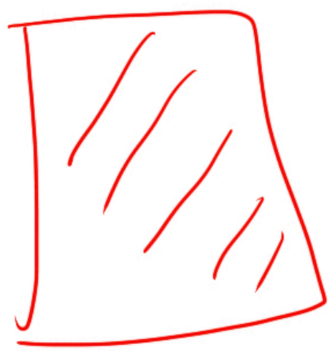
GUI

package manager

EA → enhancement advisories
BA → bugfix advisories
SA → security advisories



register with out CDN



RHEL system

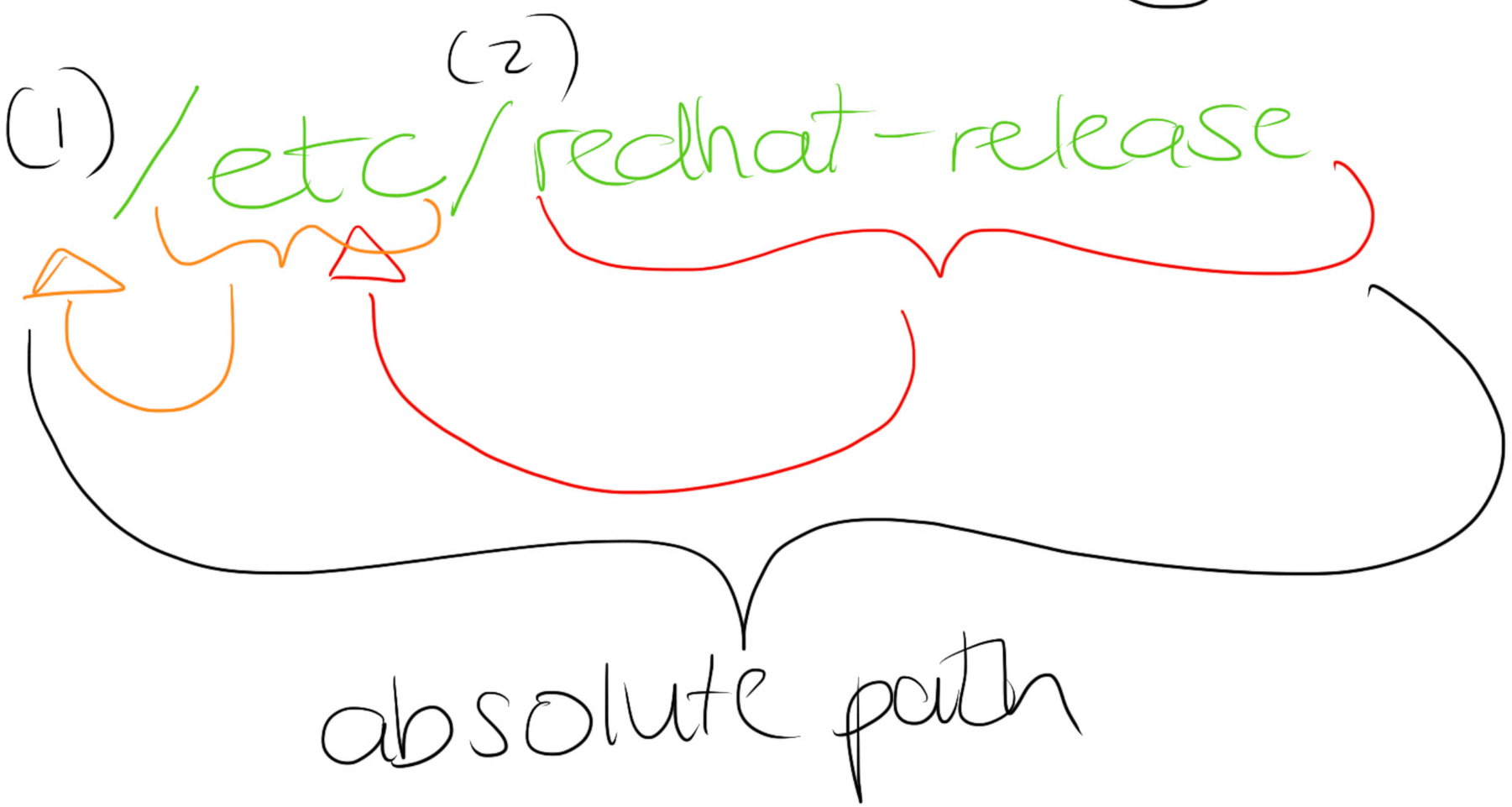
ISO file (January 2021)

Fedora → NOT SUPPORTED
Test bed
Latest in OSS developments
NOT FOR PRODUCTION USE

RHEL → IS SUPPORTED
IS STABLE
MAY not have latest versions
IS FOR PRODUCTION USE
Requires subscription

CentOS → Is not supported
Is stable
may be few releases behind RHEL
Is not for production use
Is based on RHEL

/ → The root of filesystem (1)
→ also file/directory seperator (2)



. → current directory

.. → parent directory of current directory

- → old working directory

Go through find, sudo, groups, variables
permissions, ssh keys,
network manager, tar

yum

users → uid

groups → gid

processes → pid

files → index node or inode

ricardo: uid 1000

ricardo: uid 2000

inode

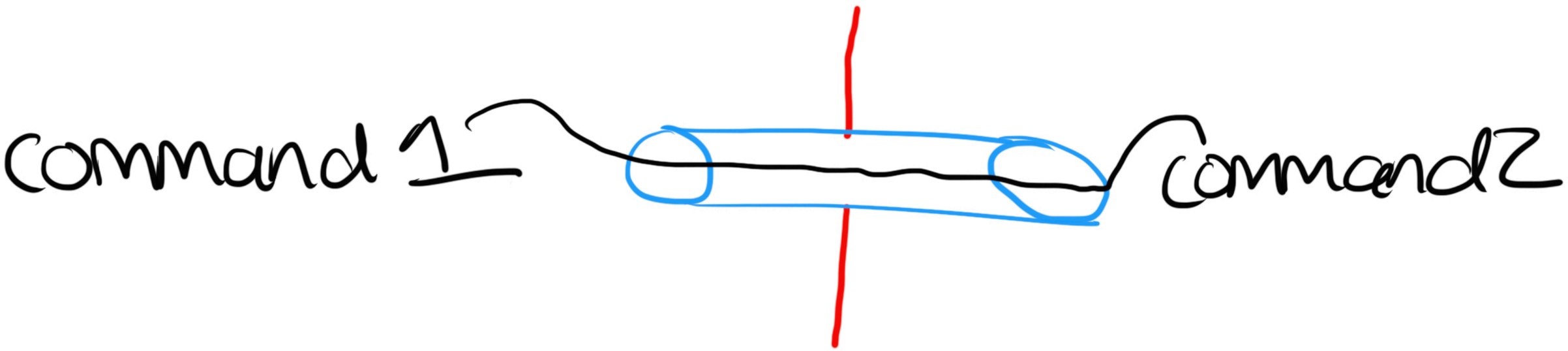
- is 128 bytes in size
- specifies permissions, ownership, date and timestamps of file
- contains panters to data blocks for file

/etc/redhat-release — 11708

8 KB → block device

2KB block size for filesystem

4 files can be created


command 1
command 2

find "starting where" "how" "what" ["and then what"]

user/group logging in from = (as these users) commands

ricardo ALL = (root) /usr/bin/passwd

! /usr/bin/passwd root

! /usr/bin/passwd -l root

rwx

① Is the user in question, the owning user of a file. If yes, stop right here and apply. If no, go to 2.

② Is the user in question, a member of the owning group associated with the file. If yes, stop right here and apply. If no, go to 3.

③ You are qualified by 'other' and the permissions for 'other' apply

ACL permissions workflow

① owning user ? ← inode

↓

② named user (as per ACL) ? ← fs metadata

↓

③ owning group ? ← inode

↓

④ named group (as per ACL) ? ← fs metadata

↓

⑤ qualified by other ← inode

Sticky bit (directories only)

chmod o+t /dir or chmod 1xyz /dir

Allows users to delete files which they own

Set Gid bit (files and directories)

chmod g+s /file or /dir

or

chmod 2xyz /file or /dir

New files inherit the owning group from parent directory, and not the creator.

For files, when a file is executed and has the setgid bit set, the executable runs with the permissions of the owning group of the file

Setuid bit (files only)

chmod u+s /file or chmod 4xyz /file

Executable runs with permissions of owning user of the file


Active Directory
getent passwd
getent group
/etc/passwd
/etc/group
/etc/shadow
local files

directories

umask

```
 777
 022
-----
 755
```

```
 777
 077
-----
 700
```

files

umask

```
 666
 022
-----
 644
```

```
 666
 077
-----
 600
```

IDM ← part of RHEL sub

└ LDAP

└ kerberos

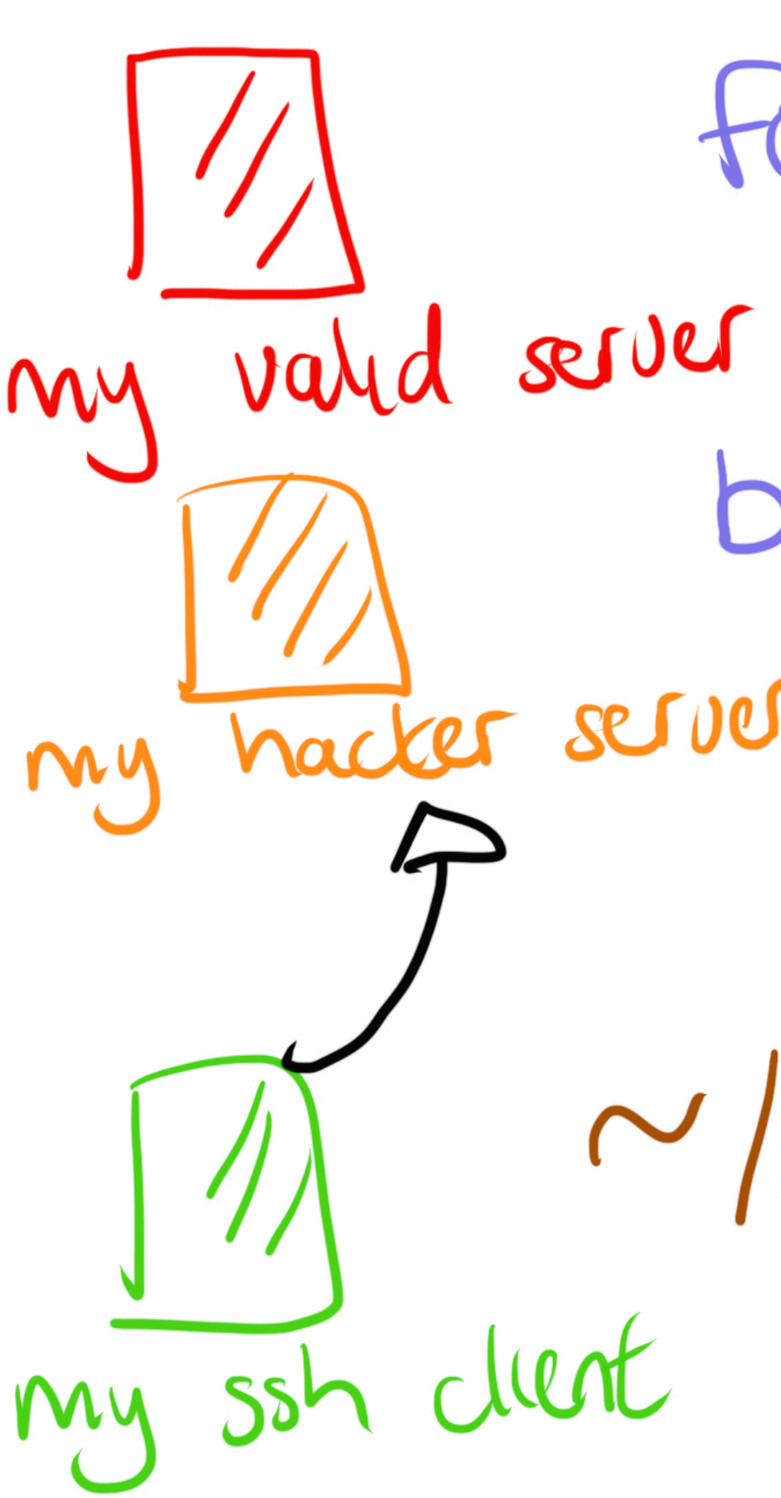
└ ...

bobby (database servers group)

IDM ← trust → Active Directory Linux admins

db db

Linux systems


foo (host key)
my valid server
bar (host key)
my hacker server
~/.ssh/known_hosts
my ssh client

systemd (pid 1)

service

socket

mount

device

target

web.target

└ nginx.service
cockpit.socket
vsftpd.socket
nfs_web.mount

facility.priority

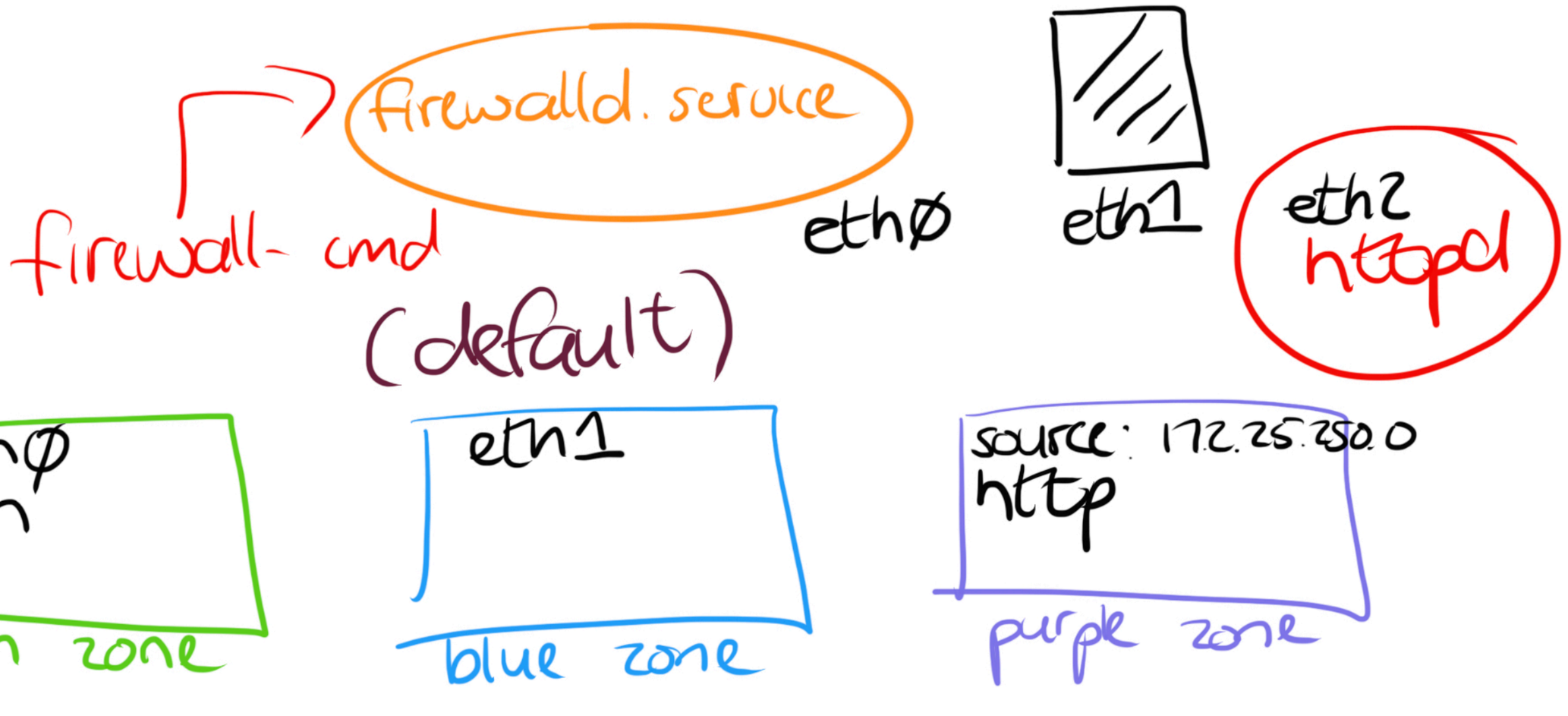
- priority: represents the urgency of a message
- facility: represents a subsystem responsible for generating a message
  - are predefined
    - developers and/or sysadmins configure services/subsystems to use a predefined facility

rsyslogd

└ uses syslog protocol

client 1
172.25.250.100

client 2
172.25.251.200



source: 172.25.250.0/24

default

rich rule: [1] not http

[2] allow all ports

a.b.c.d

blue zone

source: w.x.y.z

service: http

green zone

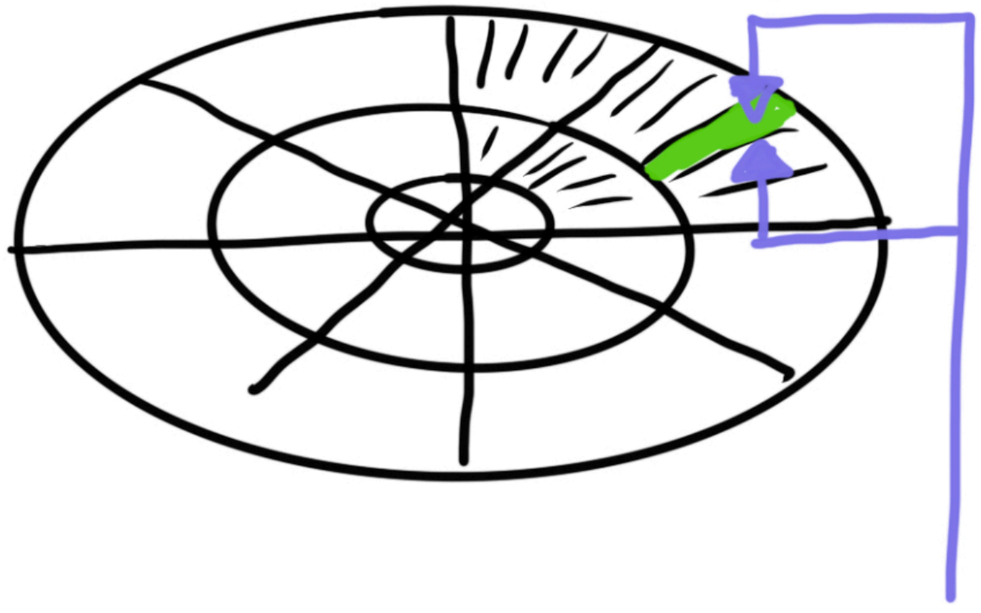
1. match to interface ?
2. match to source ?
3. default zone

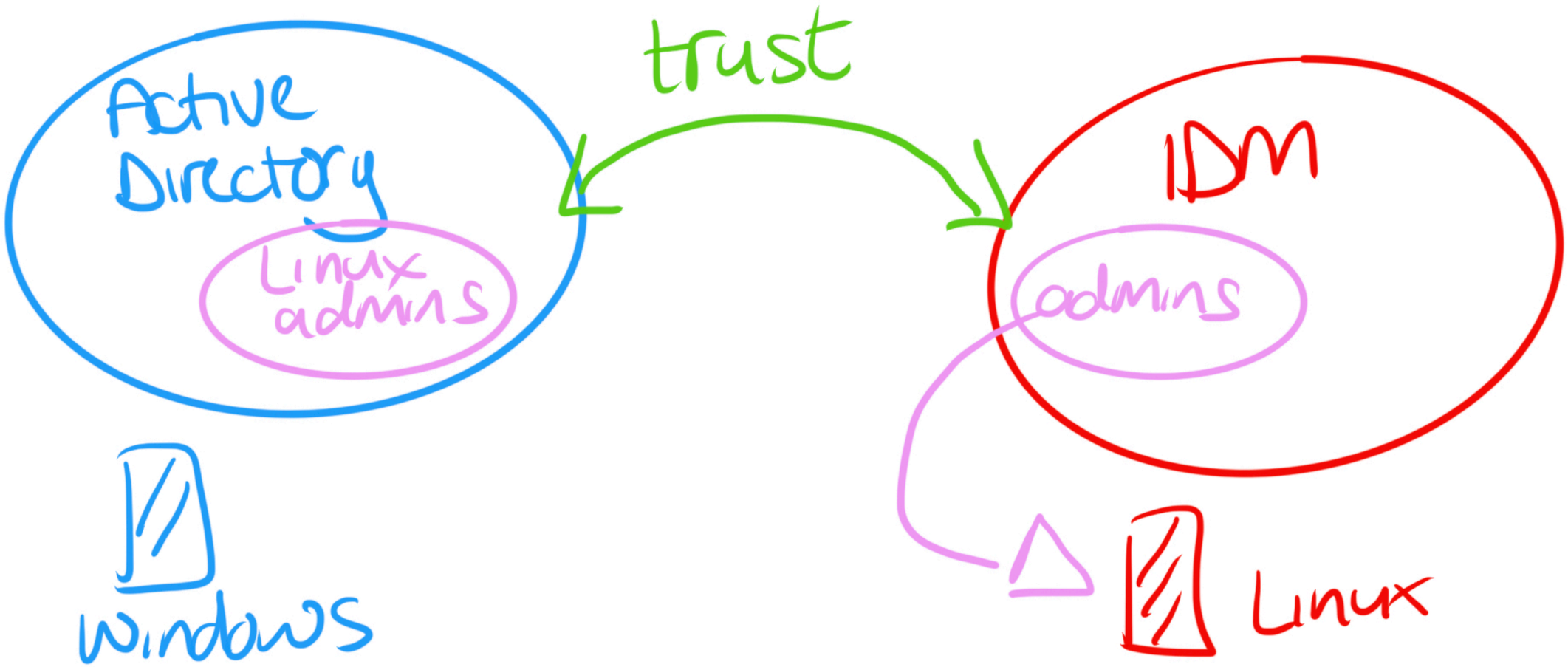
8 bits in 1 byte

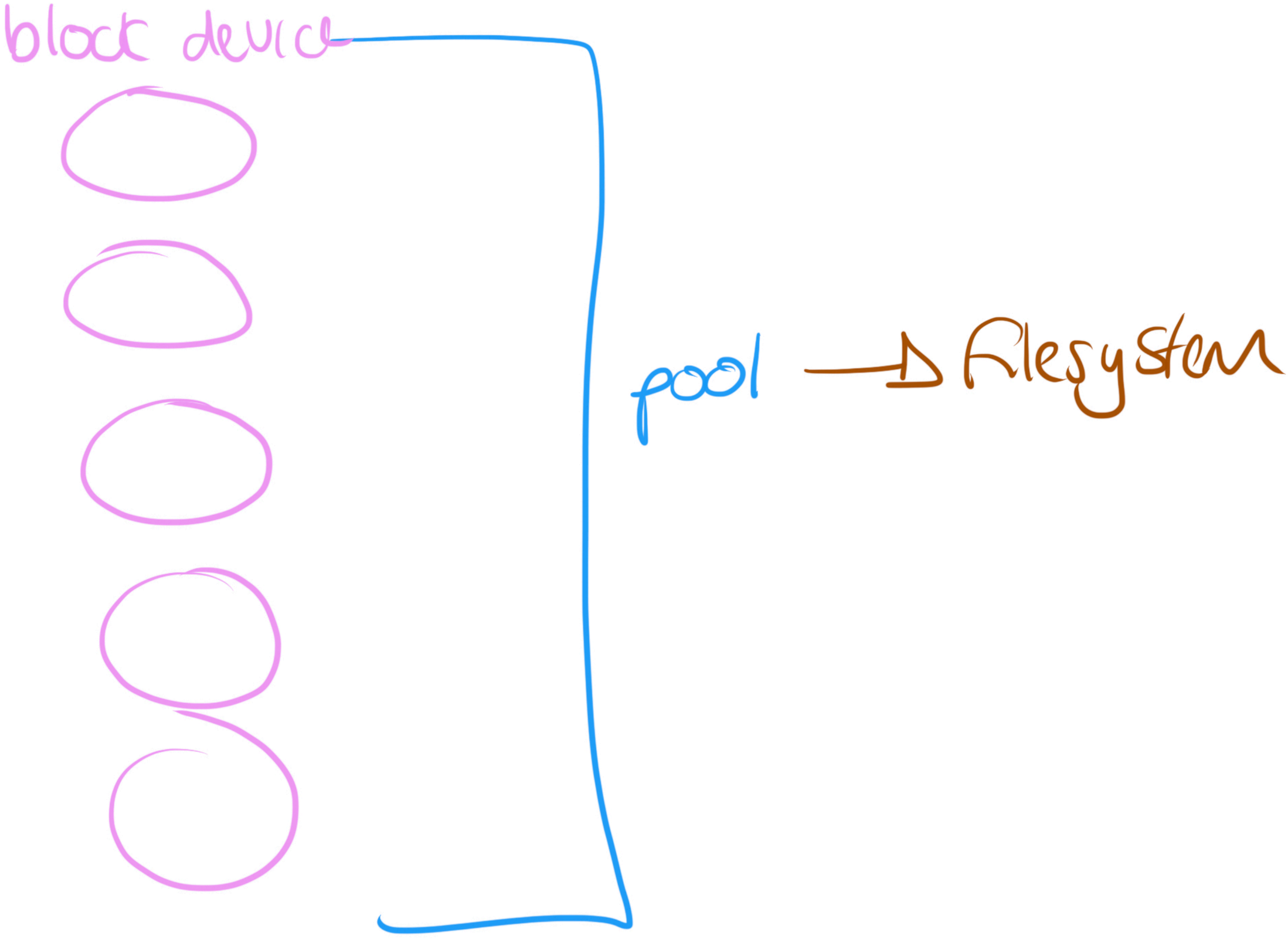
1000 bytes in 1 kilobyte (kB) 10^3

1024 bytes in 1 kibibyte (KiB) 2^{10}

1 sector : 512 bytes




trust
Active Directory
Linux admins
IDM
admins
Windows
Linux


block device
pool
filesystem

+ OS (RHEL)
+ runtime (nodejs 10)
+ application server (nodejs)
+ application (nodejs)

40% utilization

physical machine

+OS
+app runtime
+application server
+application

virtual machine

nodejs 10

kernel
grub
systemd
journald
Dbus
...

CAGE

+OS
+app runtime
+app server
+app

virtual machine

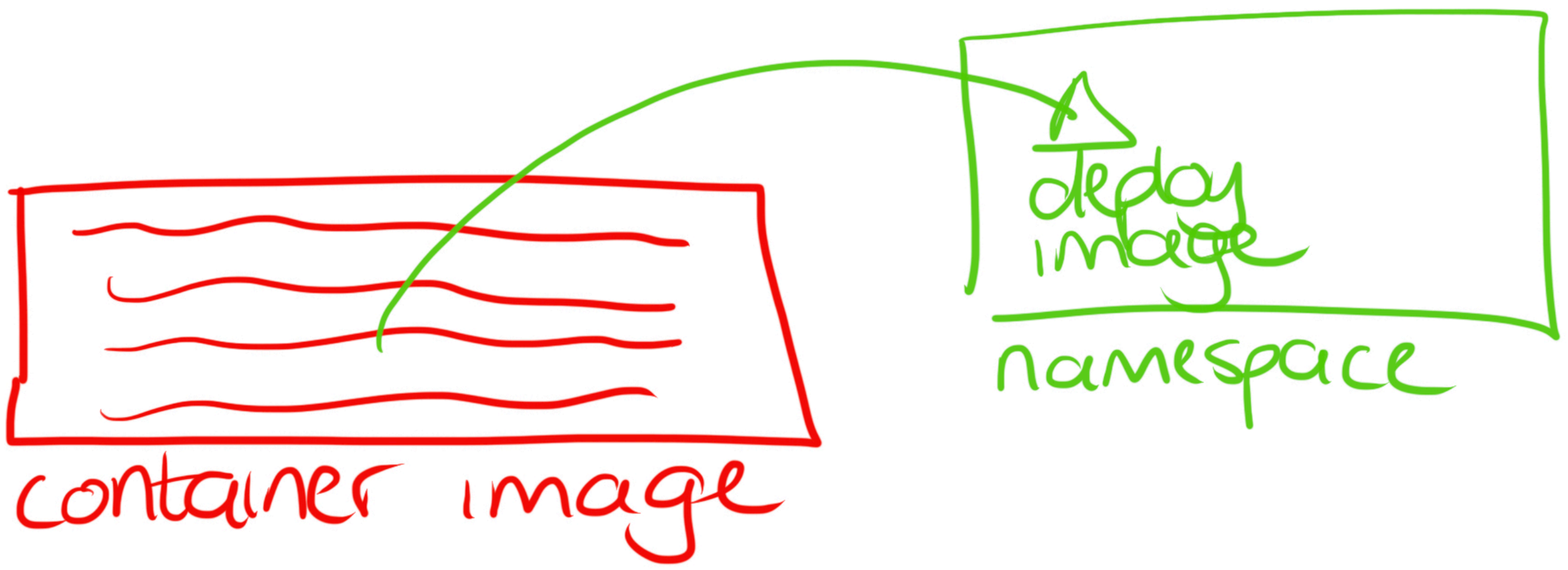
nodejs 12

List of files needed by application:

/etc/fstab
/usr/bin/systemd

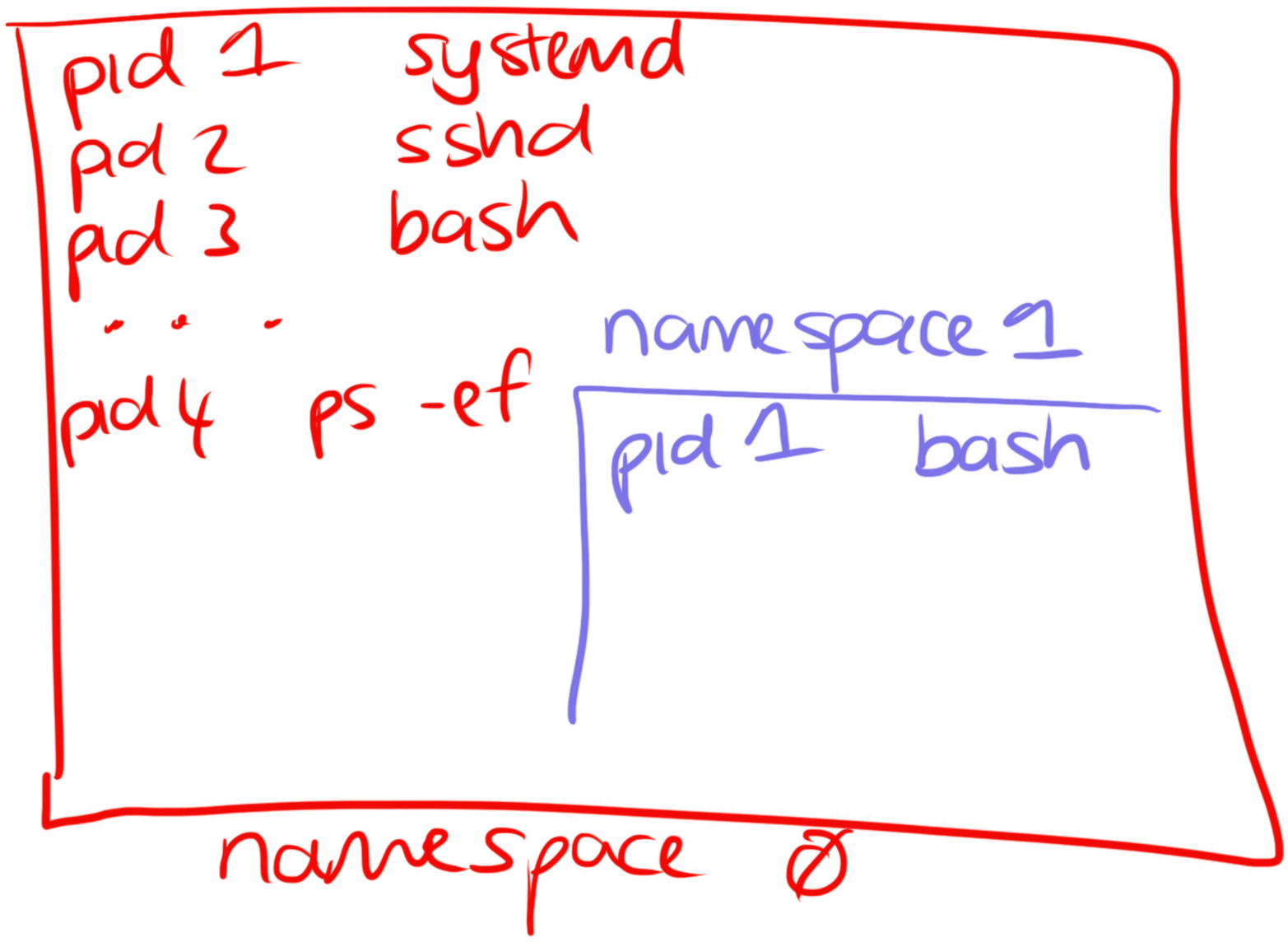
A container is a running instance of a container image.

A container image is a single file (tar ball) which contains all the files needed by your application, including the application itself.



The Linux kernel

- namespaces
- control groups
- selinux
- seccomp


pid 1 systemd
pid 2 sshd
pid 3 bash
. . .
pid 4 ps -ef
namespace 1
pid 1 bash
namespace Ø

Requirements for containerization:

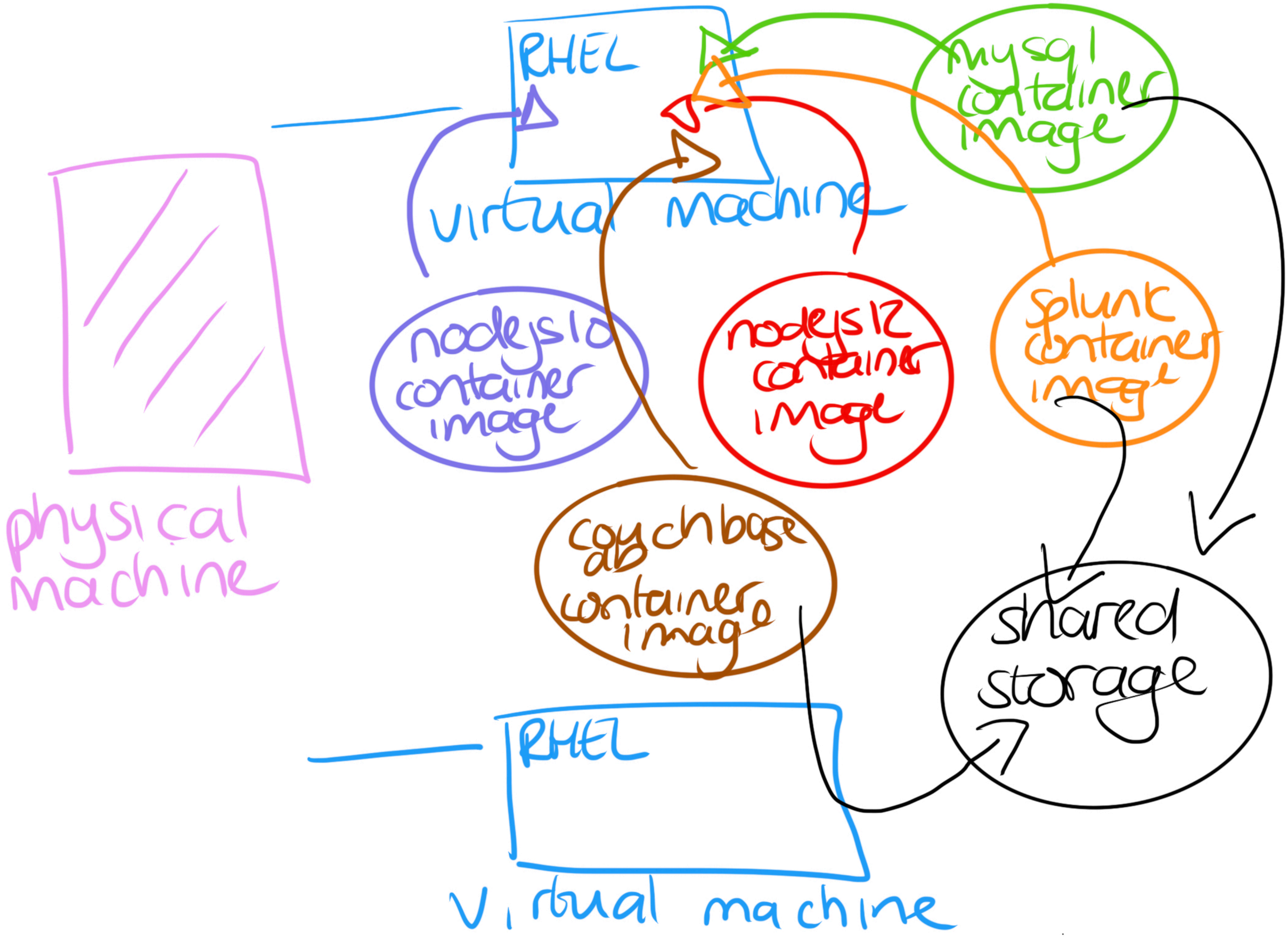
[1] container image

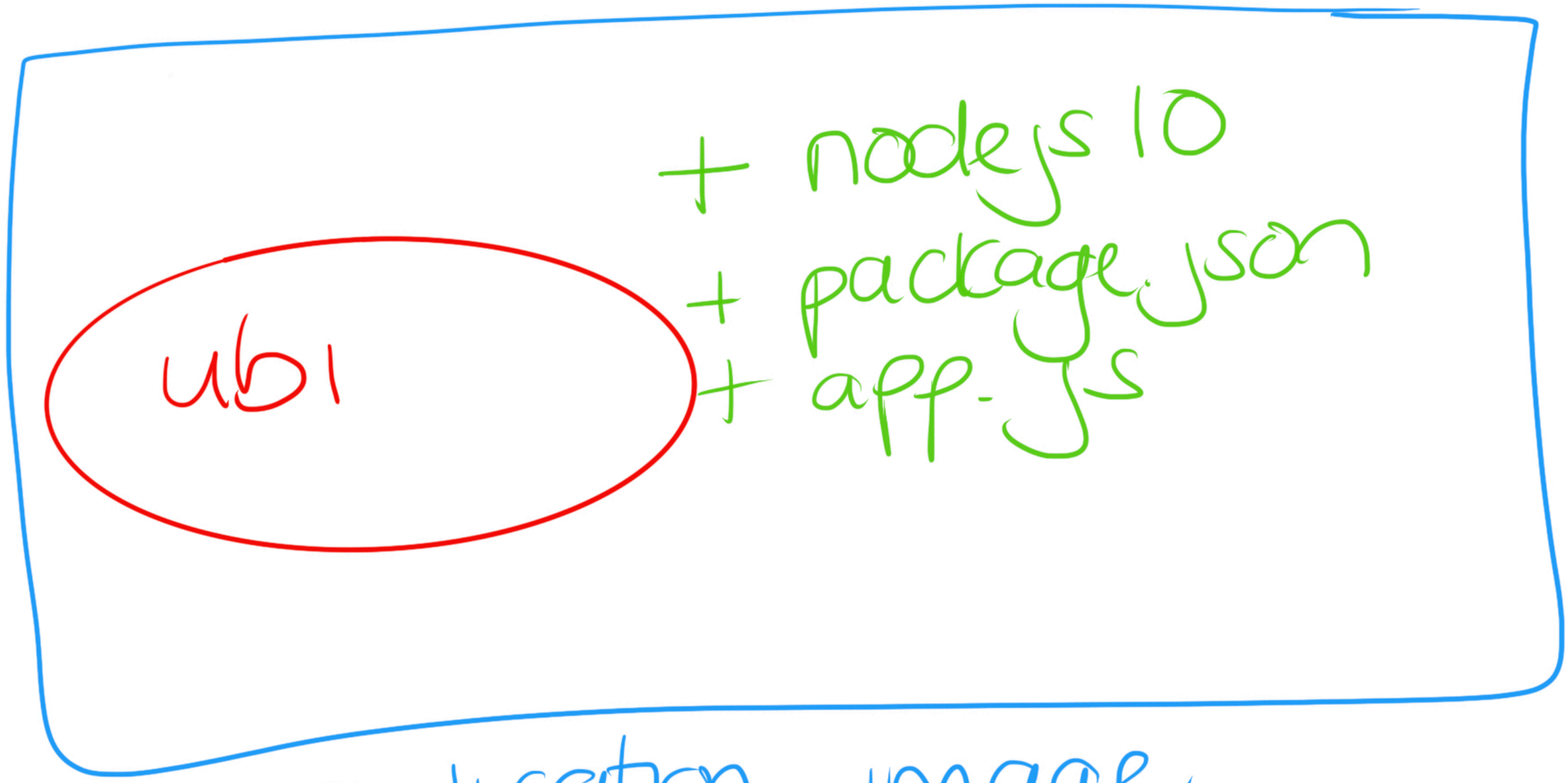
[2] container runtime

- docker
- runc (library)
- cri-o

[3] container management tool

- docker
- podman
- crictl


RHEL
virtual machine
mysql container image
nodejs10 container image
nodejs12 container image
splunk container image
couchbase db containers image
shared storage
physical machine
RHEL
Virtual machine



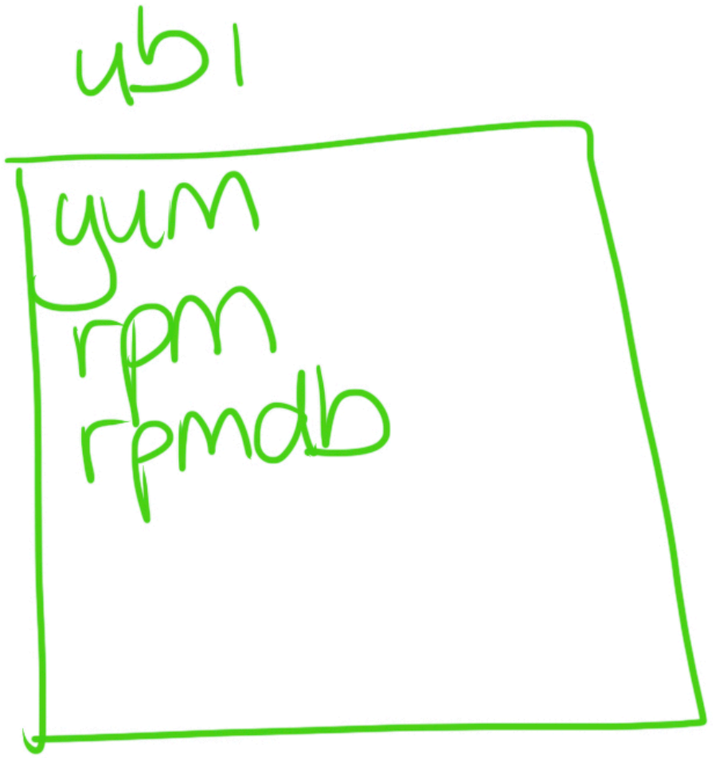
application image

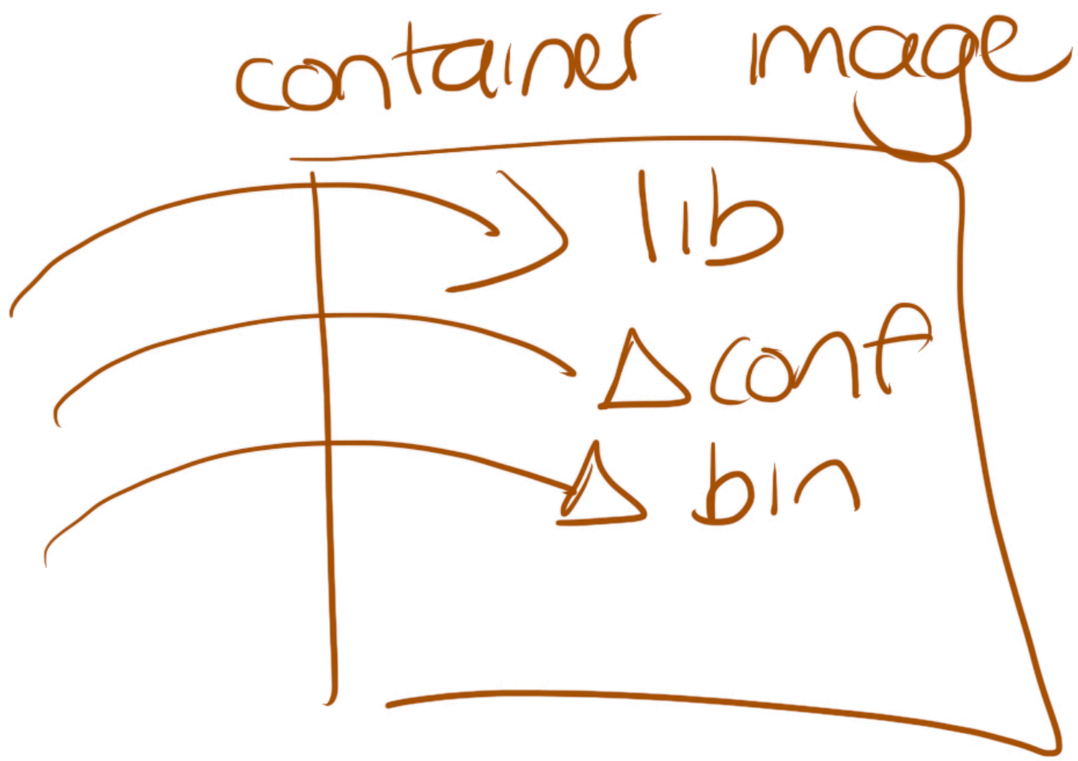
Containerfile

```
FROM ubi
RUN yum install
RUN copy app.js
```

buildah → tool to build images

buildah yum install nodejs

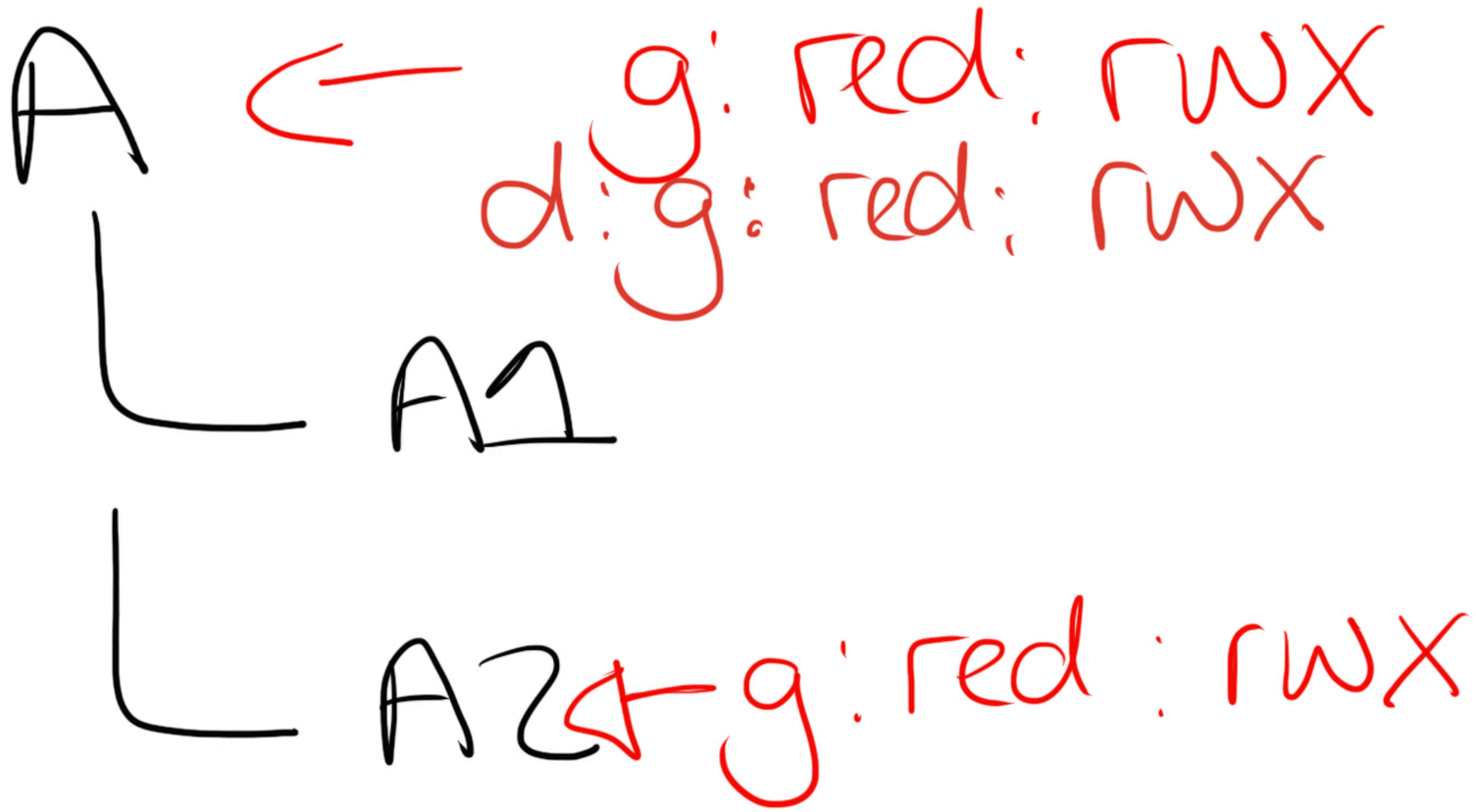


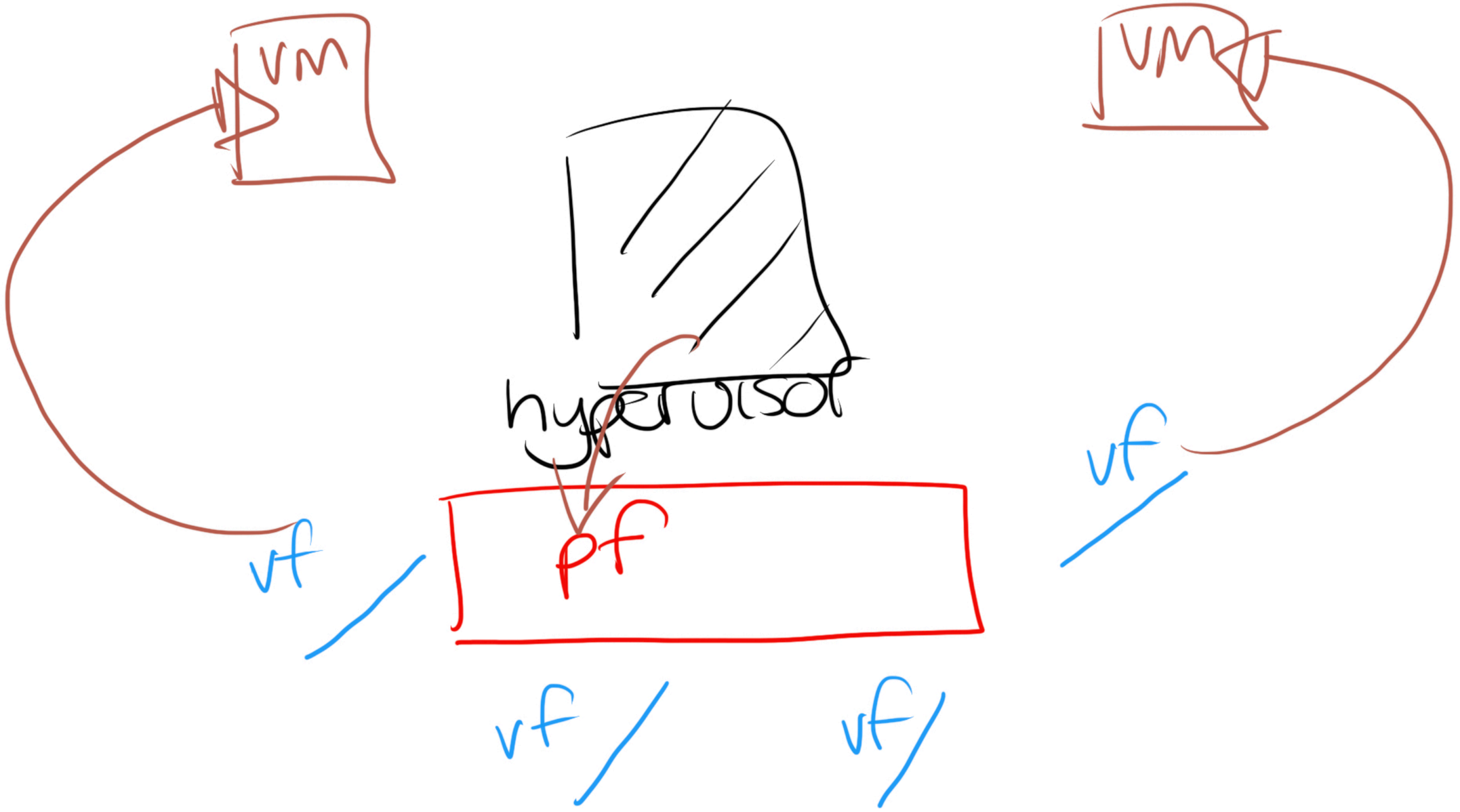


boot loader
kernel
initramfs
systemd

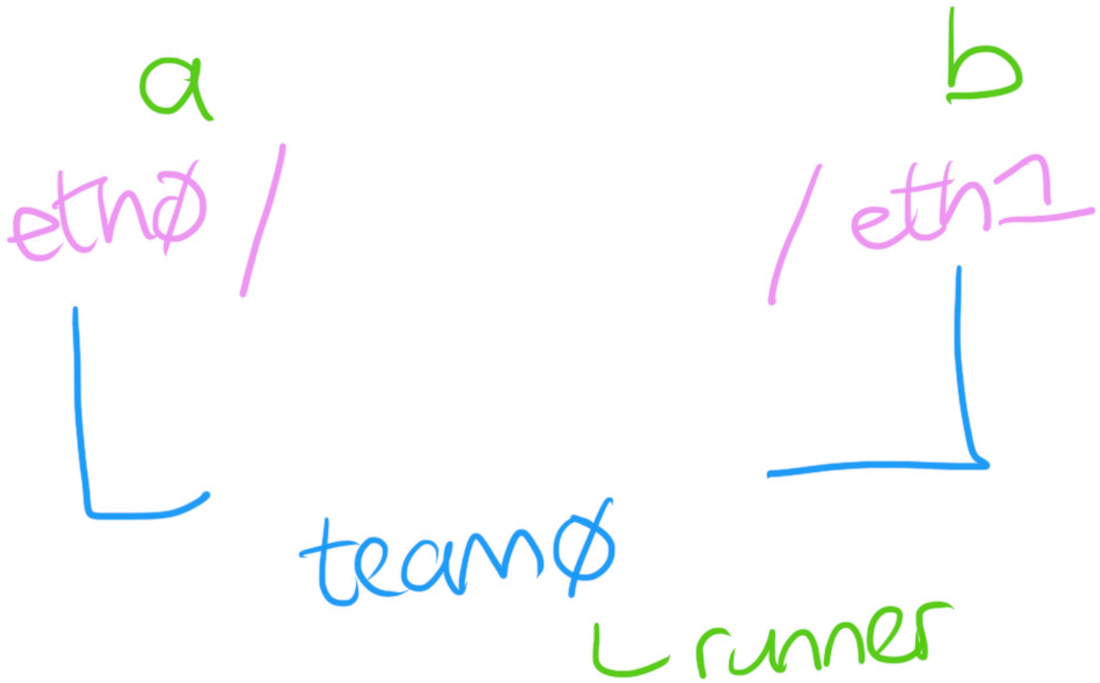
container image

kubevirt

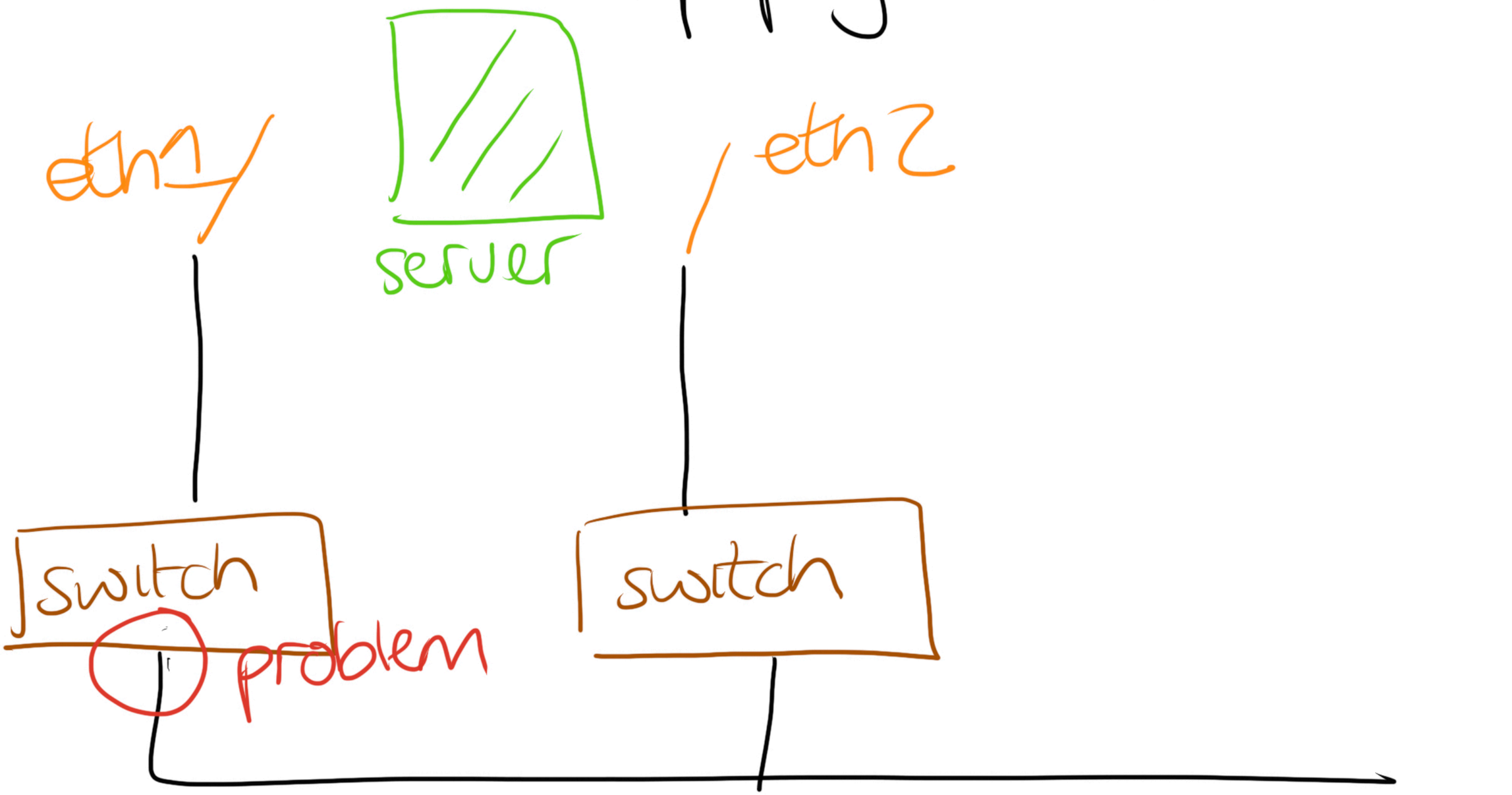

A
g: red: rwx
d: g: red: rwx
A1
A2
g: red : rwx



SR-IOV + DPDK


a
b
eth0
eth1
team0
runner

arp ping → link watch



highly available node