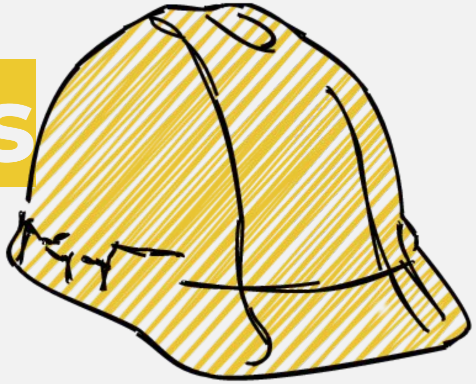
Buildroot

Making Tiny Linux Easy

Docker Containers

Hello!

I am Alexis Facques

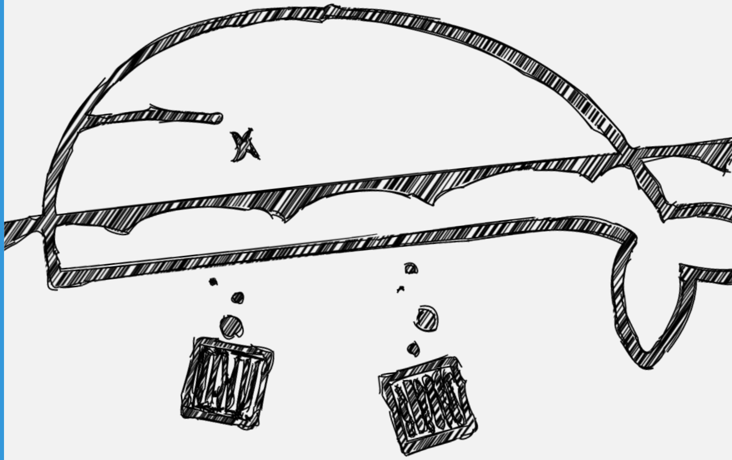


I love tiny Docker images and automating things. and Golf

Find these slides on github.com/alexisfacques

1.

A Docker story



Where it all begins...

> DevOps is a software engineering culture and practice of putting horrors into containers and then talking about Kubernetes at conferences

seasonally affected server
@sadserver

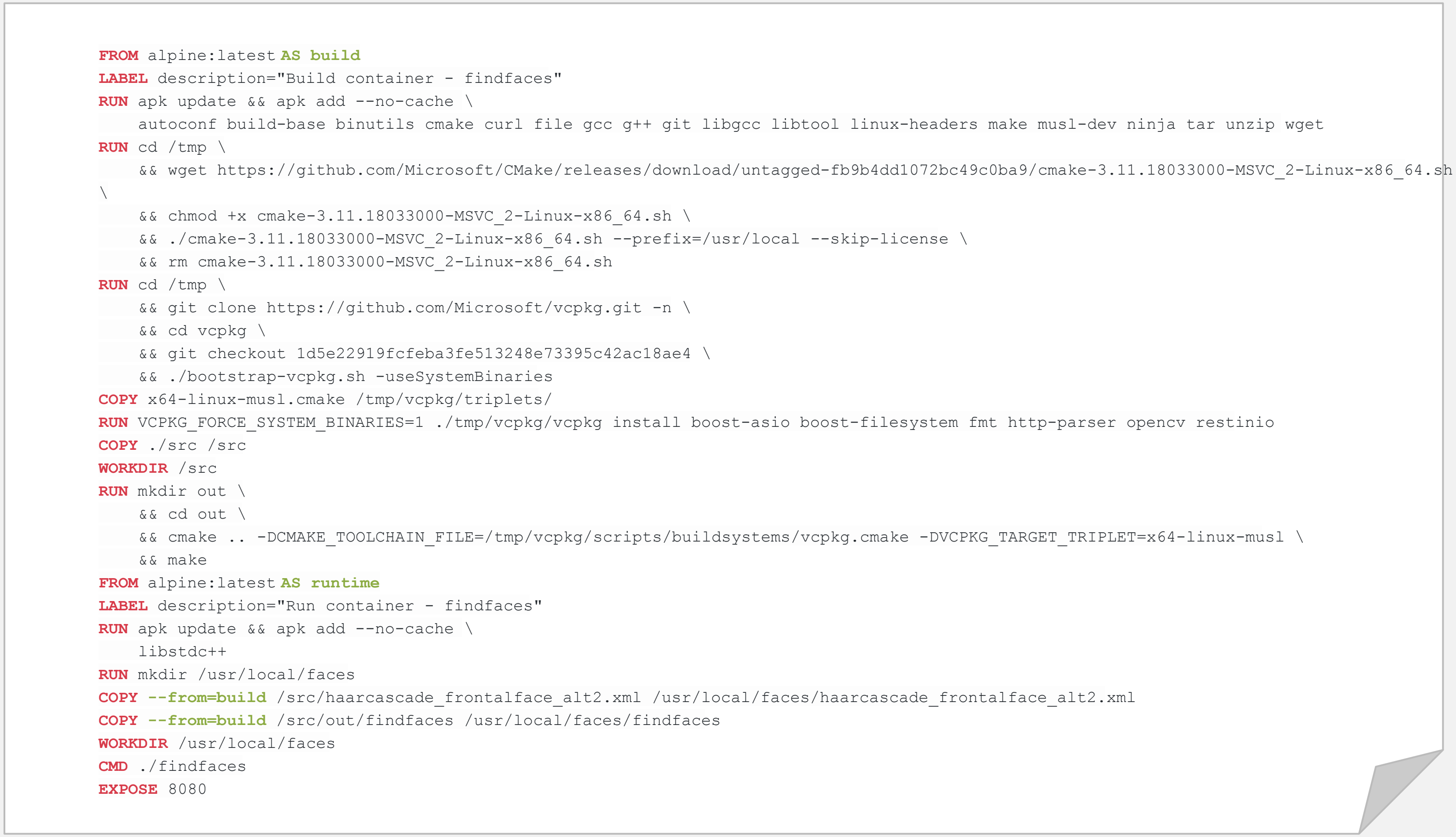
A Docker Story: The good

```
# build stage
FROM golang:alpine AS build
ADD . /src
RUN cd /src && go build -o myapp


# final stage
FROM alpine AS runtime
WORKDIR /app
COPY --from=build /src/myapp /app/
CMD ./myapp
```

A Docker Story: The bad

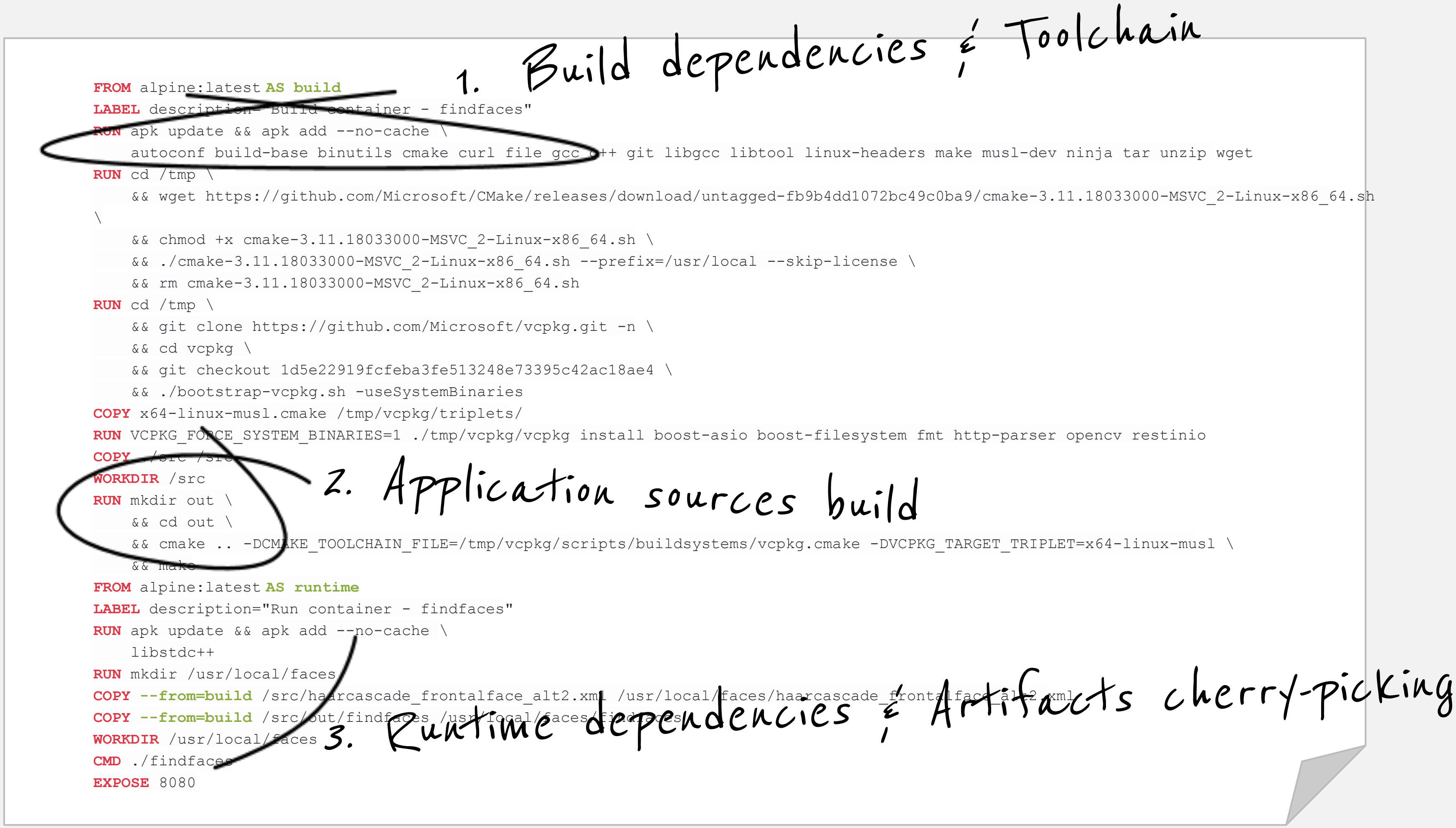
```
FROM ubuntu:18.04

RUN apt-get update && apt-get upgrade -y && apt-get install cppcheck libgecode-dev g++ cmake libboost-all-dev git wget unzip -y

COPY . /src
RUN cd /src && cmake ./CMakeLists.txt && make -j10

WORKDIR /src/bin
CMD ./myapp
```

A Docker Story: The ugly



```
FROM alpine:latest AS build
LABEL description="Build container - findfaces"
RUN apk update && apk add --no-cache \
    autoconf build-base binutils cmake curl file gcc g++ git libgcc libtool linux-headers make musl-dev ninja tar unzip wget
RUN cd /tmp \
    && wget https://github.com/Microsoft/CMake/releases/download/untagged-fb9b4dd1072bc49c0ba9/cmake-3.11.18033000-MSVC_2-Linux-x86_64.sh
\
    && chmod +x cmake-3.11.18033000-MSVC_2-Linux-x86_64.sh \
    && ./cmake-3.11.18033000-MSVC_2-Linux-x86_64.sh --prefix=/usr/local --skip-license \
    && rm cmake-3.11.18033000-MSVC_2-Linux-x86_64.sh
RUN cd /tmp \
    && git clone https://github.com/Microsoft/vcpkg.git -n \
    && cd vcpkg \
    && git checkout 1d5e22919fcfeba3fe513248e73395c42ac18ae4 \
    && ./bootstrap-vcpkg.sh -useSystemBinaries
COPY x64-linux-musl.cmake /tmp/vcpkg/triplets/
RUN VCPKG_FORCE_SYSTEM_BINARIES=1 ./tmp/vcpkg/vcpkg install boost-asio boost-filesystem fmt http-parser opencv restinio
COPY ./src /src
WORKDIR /src
RUN mkdir out \
    && cd out \
    && cmake .. -DCMAKE_TOOLCHAIN_FILE=/tmp/vcpkg/scripts/buildsystems/vcpkg.cmake -DVCPKG_TARGET_TRIPLET=x64-linux-musl \
    && make
FROM alpine:latest AS runtime
LABEL description="Run container - findfaces"
RUN apk update && apk add --no-cache \
    libstdc++
RUN mkdir /usr/local/faces
COPY --from=build /src/haarcascade_frontalface_alt2.xml /usr/local/faces/haarcascade_frontalface_alt2.xml
COPY --from=build /src/out/findfaces /usr/local/faces/findfaces
WORKDIR /usr/local/faces
CMD ./findfaces
EXPOSE 8080
```

A Docker Story: The ugly



1. Build dependencies & Toolchain

```
FROM alpine:latest AS build
LABEL description="Build container - findfaces"
RUN apk update && apk add --no-cache \
    autoconf build-base binutils cmake curl file gcc g++ git libgcc libtool linux-headers make musl-dev ninja tar unzip wget
RUN cd /tmp \
    && wget https://github.com/Microsoft/CMake/releases/download/untagged-fb9b4dd1072bc49c0ba9/cmake-3.11.18033000-MSVC_2-Linux-x86_64.sh
\
    && chmod +x cmake-3.11.18033000-MSVC_2-Linux-x86_64.sh \
    && ./cmake-3.11.18033000-MSVC_2-Linux-x86_64.sh --prefix=/usr/local --skip-license \
    && rm cmake-3.11.18033000-MSVC_2-Linux-x86_64.sh
RUN cd /tmp \
    && git clone https://github.com/Microsoft/vcpkg.git -n \
    && cd vcpkg \
    && git checkout 1d5e22919fcfeba3fe513248e73395c42ac18ae4 \
    && ./bootstrap-vcpkg.sh -useSystemBinaries
COPY x64-linux-musl.cmake /tmp/vcpkg/triplets/
RUN VCPKG_FORCE_SYSTEM_BINARIES=1 ./tmp/vcpkg/vcpkg install boost-asio boost-filesystem fmt http-parser opencv restinio
```

2. Application sources build

```
COPY ./src /src
WORKDIR /src
RUN mkdir out \
    && cd out \
    && cmake .. -DCMAKE_TOOLCHAIN_FILE=/tmp/vcpkg/scripts/buildsystems/vcpkg.cmake -DVCPKG_TARGET_TRIPLET=x64-linux-musl \
    && make
```

3. Runtime dependencies & Artifacts cherry-picking

```
FROM alpine:latest AS runtime
LABEL description="Run container - findfaces"
RUN apk update && apk add --no-cache \
    libstdc++
RUN mkdir /usr/local/faces
COPY --from=build /src/haarcascade_frontalface_alt2.xml /usr/local/faces/haarcascade_frontalface_alt2.xml
COPY --from=build /src/out/findfaces /usr/local/faces/findfaces
WORKDIR /usr/local/faces
CMD ./findfaces
EXPOSE 8080
```

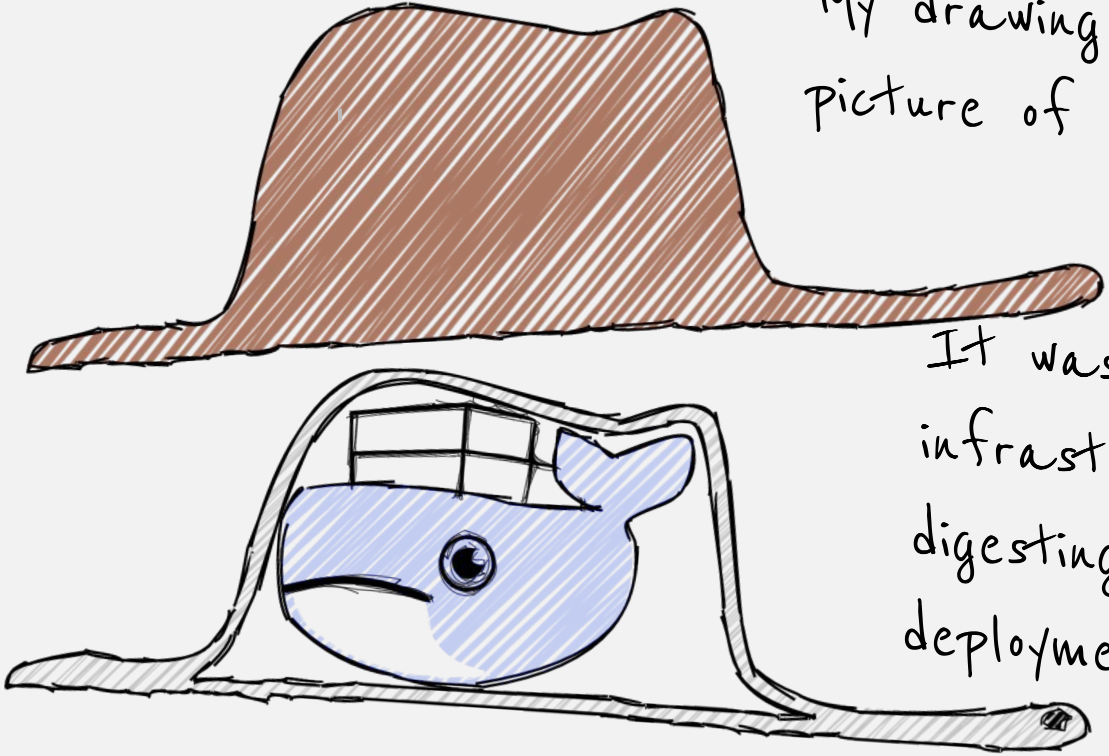
2.

Size matters

... Does it ?

Size matters: When orchestrating containers

- A “production” Kubernetes cluster often means shared registry, distributed storage & heavy internal traffic.



My drawing was not a picture of a hat.

It was a picture of my infrastructure network digesting my DaemonSet deployment.

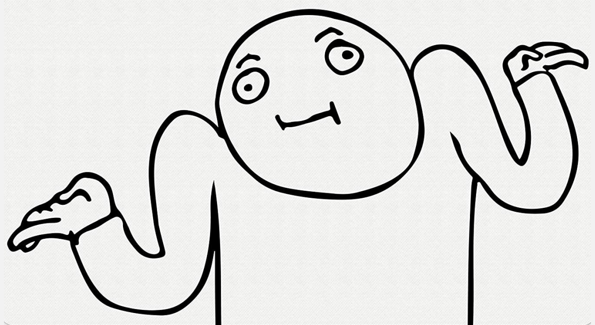
Size matters: When orchestrating containers

- A production Kubernetes cluster induces shared registry, storage & heavy internal traffic.

- Bigger are your images, longer are your deployments;

- A “generally accepted” solution:

  Prepulls & pods affinity.



194MB

Glibc Debian Jessie

129MB

Glibc Ubuntu 16:04

5MB

Musl Alpine

Basically a containerized package manager

Size matters: Security by design

- Top two most popular Docker images have over 500 vulnerabilities;

- 80% developers are not addressing containers security.

567

Vulnerabilities in node:latest

65

Vulnerabilities in node:10-slim

0

Vulnerabilities in node:10-alpine

3.

Embedded systems & Buildroot

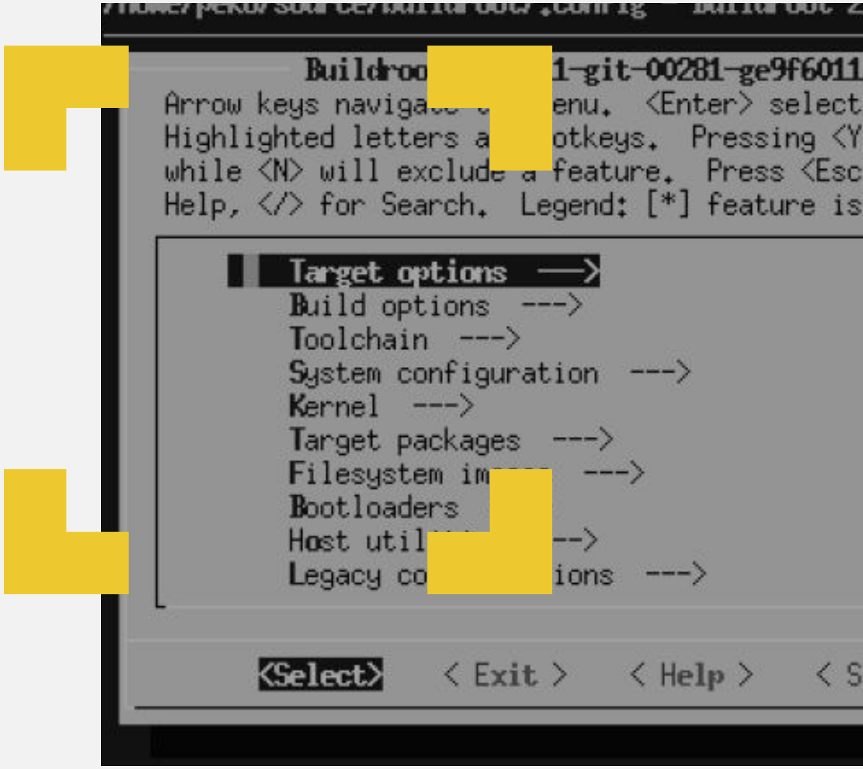
Let's get into the action

Buildroot

- Free, open-source, well-maintained;
- Cross-compilation tool;
- Build from scratch from source;
- Pick your architecture, filesystem & packages; Buildroot will output a minimized rootfs.

Buildroot: In a nutshell

- Easy feature selection with kconfig;
- Hard to learn, easy to master;
- BR2_EXTERNAL keeps your customizations outside of Buildroot;

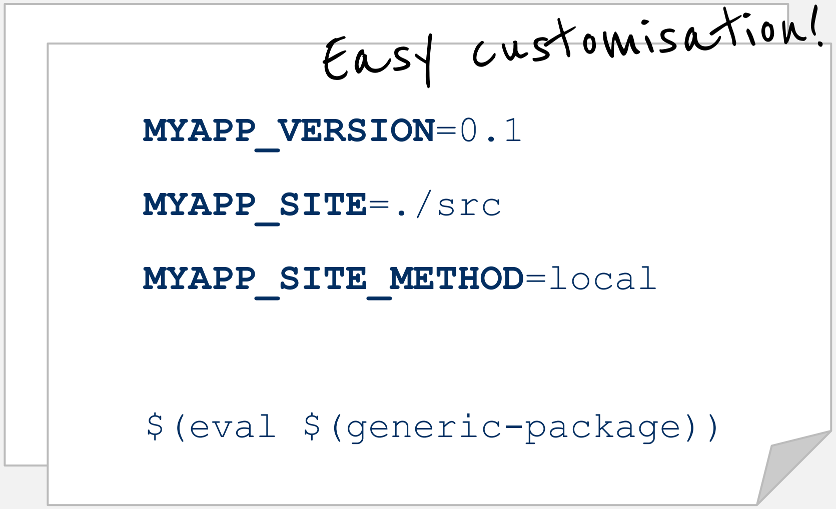


```
/home/peko/source/buildroot/.config - Buildroot 2
 Buildroo        1-git-00281-ge9f6011
 Arrow keys naviga     enu.  <Enter> select
 Highlighted letters a   otkeys.  Pressing <Y
 while <N> will exclude a feature.  Press <Esc
 Help, </> for Search.  Legend: [*] feature is

        Target options  --->
        Build options  --->
        Toolchain  --->
        System configuration  --->
        Kernel  --->
        Target packages  --->
        Filesystem im      --->
        Bootloaders
        Host util      --->
        Legacy co      ions  --->

    <Select>    < Exit >    < Help >    < S
```

Buildroot: Demo time ?

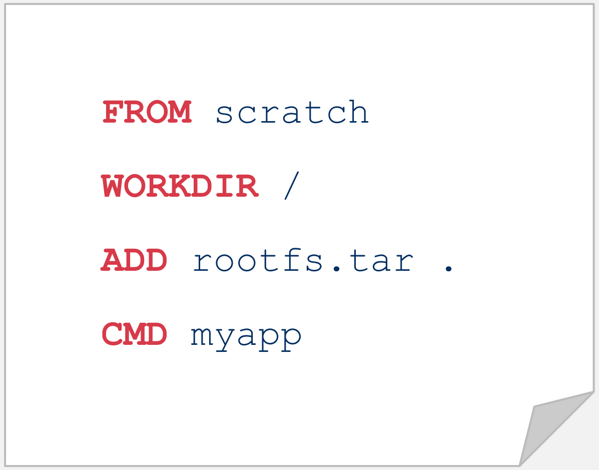
Buildroot: How it pairs with Docker

- Pros:



Easy customisation!

```
MYAPP_VERSION=0.1

MYAPP_SITE=./src

MYAPP_SITE_METHOD=local


$(eval $(generic-package))
```



```
FROM scratch

WORKDIR /

ADD rootfs.tar .

CMD myapp
```

- “Cons”:
  - Long building times (has to compile all your target toolchain from scratch);
  - Single-layer images.

77.8MB

node:alpine

19.3MB

Buildroot-based image

Who needs a package manager anyway

```
BR2_x86_64=y
BR2_TOOLCHAIN_BUILDROOT_MUSL=y
BR2_TOOLCHAIN_BUILDROOT_CXX=y
BR2_PACKAGE_NODEJS=y
```

~75%

Smaller !

Buildroot: There’s even more to see!

- 1800+ packages maintained & updated every two months;
- CVE analysis scripts per configuration file;
- Top-tier documentation;
- Additional overlays for embedded systems:
  - Raspberry Pi, Pi 3, Broadcom, generic ARM…

Thanks!

Any questions?