Linux Distro Tools for Building Container Images

Nisha Kumar (@nishakmr)

Joshua Lock (@hi_joshuagl)

Open Source Software Engineers / OSTC

@vmwopensource

November 2019

vmware

Agenda

Motivations and context

Motivations for exploring other container image build tools

Context and level setting

Investigation

Linux distros and tools

Experiments and analysis

Summary

Takeaways

Actionable learning

Motivations and context

Why are we looking at distro tools?

Motivations

Compliance

Security

Discipline

Level setting

Containers ~= packaging format

Desirable properties of a container image

...or package format

Repeatable

At any given time, we can reliably create an equivalent image from source

Identifiable

We can reason about their contents

Recent

Contents are not old and vulnerable

Common practices which violate the desirable properties

Performing non-deterministic operations (RUN)

Encoding build-time dependent state in the image (apt-get update)

Inserting untraceable files into the filesystem (ADD/COPY)

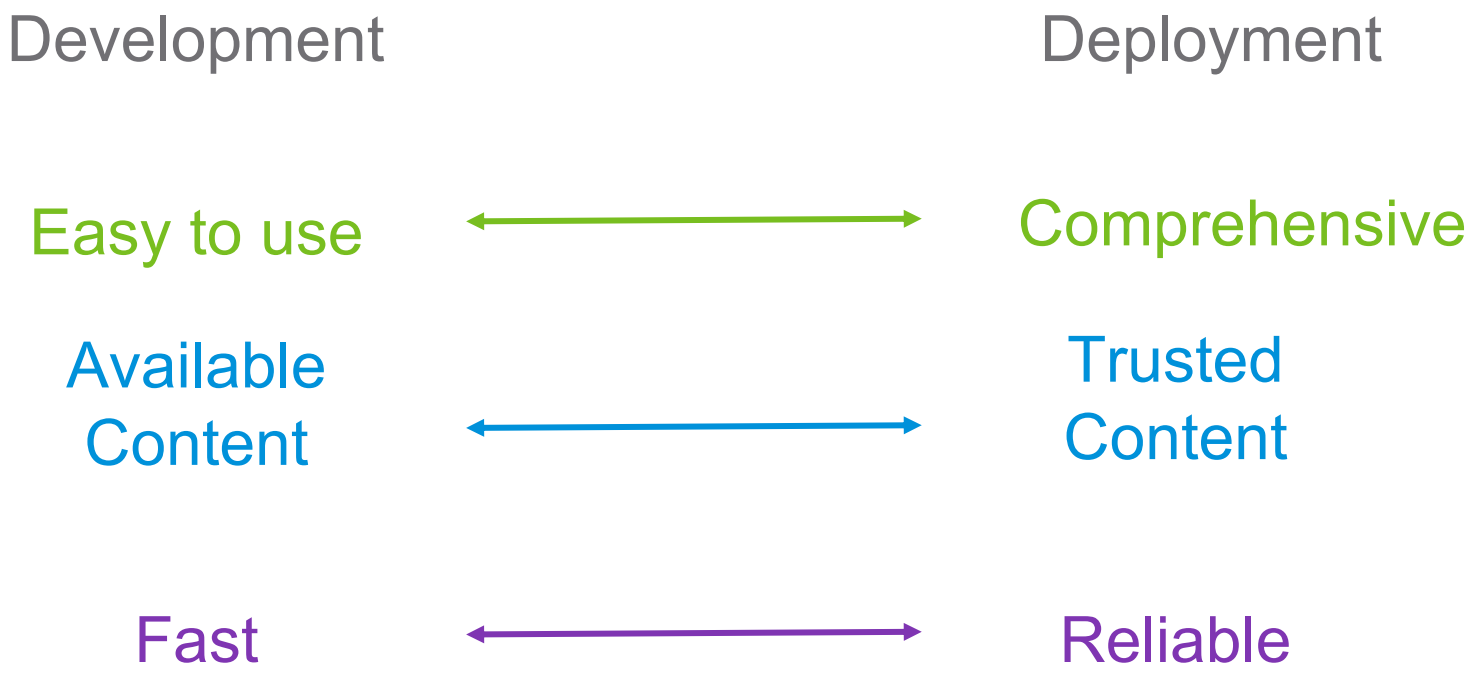
Using an old base OS (FROM)

These problems have been solved before

Why reinvent the wheel?

Tool usability continuum

...or DevOps



Investigation

Using distribution tools to build container images

Linux distributions and tools

Choosing is hard

Debian and DebOS

- Debian – the universal OS
- DebOS – Debian image generator created for Embedded Linux use-cases

Buildroot

- Embedded Linux image creator

Yocto Project

- Embedded Linux distribution builder

Guix SD

- Functional distro built on Guix package manager, inspired by Nix
- Scheme extensions provide DSLs for packaging and configuration

Distribution tool showdown

Trying to use the tools and compare the outputs

Compare

- Ease of use
- Output image size
- Engineering effort
- Quality/presence of SBoM

Experiment 1: base OS for Go applications

Base OS with Go toolchain

Goal: create a container image with Go toolchain

DebOS:

- Ease of use 😬
- Image size 😌
- Engineering effort 😎
- SBoM 😕

Experiment 1: base OS for Go applications

Base OS with Go toolchain

Goal: create a container image with Go toolchain

Buildroot:

- Ease of use 😢
- Image size ??
- Engineering effort 😖
- SBoM 😯

Experiment 1: base OS for Go applications

Base OS with Go toolchain

Goal: create a container image with Go toolchain

Yocto Project:

- Ease of use 😫
- Image size 😃
- Engineering effort 😩
- SBoM 🤩

Experiment 1: base OS for Go applications

Base OS with Go toolchain

Goal: create a container image with Go toolchain

Guix:

- Ease of use 😖
- Image size 😤
- Engineering effort 😑
- SBoM 😕

Status Quo -> Just Enough

Current	Ideal
Base OS	App
+	+
App SDK	Dependencies
+	
App	

Packaging

Package: describe a piece of software, its dependencies and how to build/install it

Package manager: handles installation and removal of packages (and dependencies)

Can we leverage the package manager to install *only* what's *required* into an image?

Experiment 2: Just enough OS (JeOS)

Most container workloads don’t need a full OS root filesystem

Goal: create minimal container image with only the containerised app and its required dependencies.

Packaging for our target distro tools

My application isn't packaged

Each distro ecosystem has different packaging and metadata formats and tools to process them.

Tools exist to automate packaging for many Linux distributions and tools

- Their support for cloud native languages and frameworks (Go, Bazel, Java, etc) is patchy to non-existent

Automatic packaging tools:

Debian — dh-make / dh-make-golang / gem2deb / pypi2deb
Buildroot — docs for packaging for each ecosystem
Yocto Project — recipetool
Guix — guix import

What if we *could* use the native package?

Eg: Postgresql

Current

Base OS

+

Dependencies

+

App

Ideal

App

+

Dependencies

Summary

Closing remarks

Takeaways

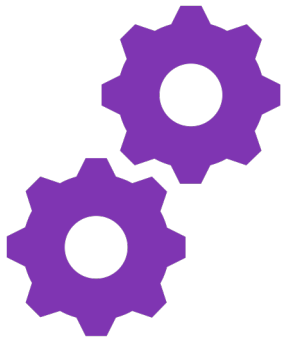
This is only a partially solved problem

Tools build for different hardware now need to be adapted for different apps

Projects are working on making the jump to cloud

Compelling reasons to help them (Open Source!)

- Documentation
- Tests
- Use cases



Actionable learning

What can we do today for my Dockerfile built containers?

Introspect your containers: Dive, Container-diff, Tern

Get to halfway

- Fairly straightforward to use debos
  - Make your Base OS + SDK with debos
  - Use this Base OS for containerization
- Glue code needed for Guix
  - Make the appropriate rootfs from gnu/store
  - Upstream above improvements to make it seamless

Resources

DebOS: https://github.com/go-debos/debos

Debian packaging: https://wiki.debian.org/Packaging/Intro?action=show&redirect=IntroDebianPackaging

Buildroot user manual: https://buildroot.org/downloads/manual/manual.html

Guix user manual: https://guix.gnu.org/manual/en/html_node/

Yocto Project Mega Manual:

https://www.yoctoproject.org/docs/3.0/mega-manual/mega-manual.html

Our Lab Notebook: https://github.com/nishakm/distro-tools-for-containers

Thank You

Joshua: @hi_joshuagl

Nisha: @nishakmr

vmware

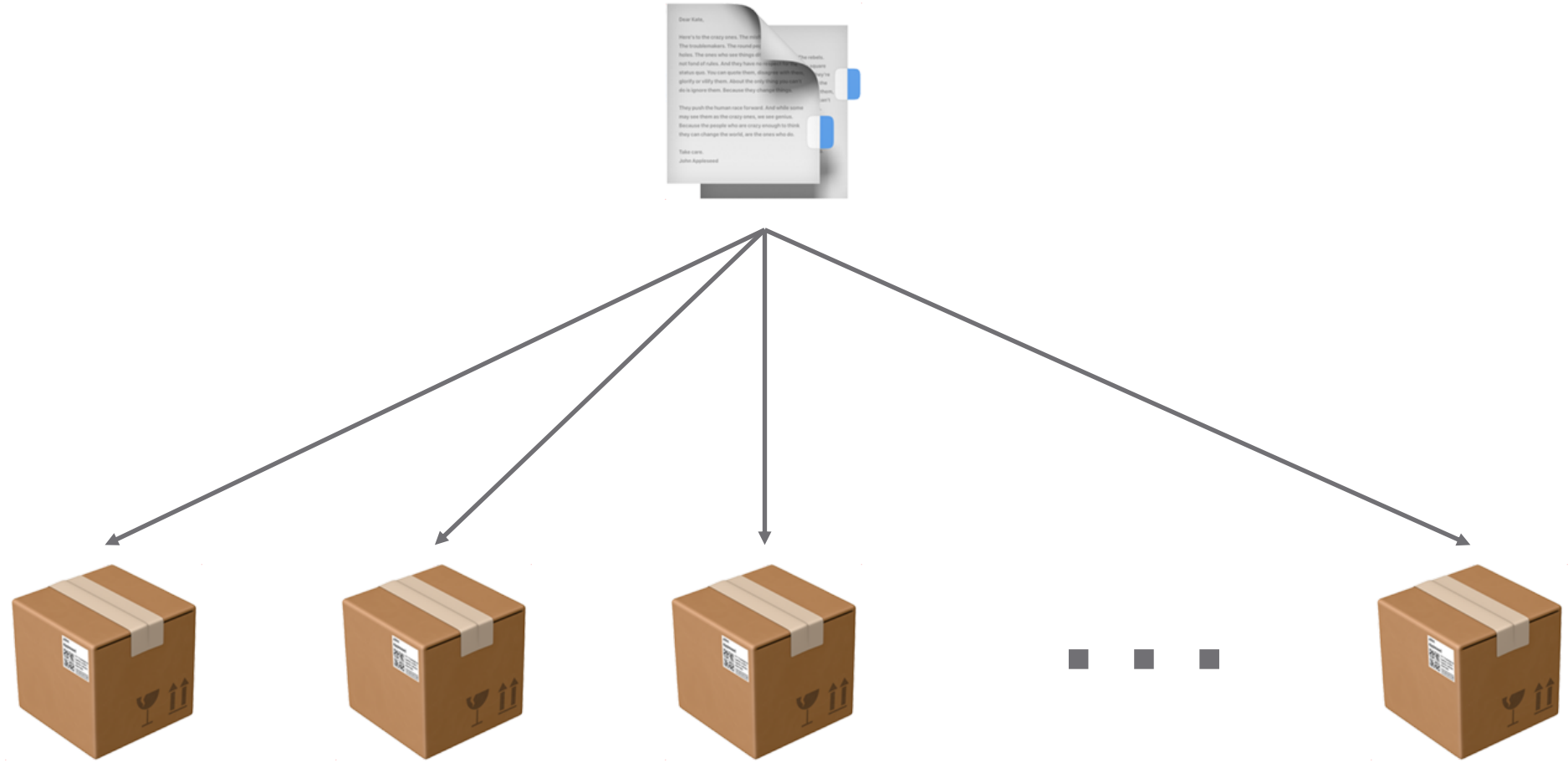
Sidebar: Image layers

Cache invalidation, image size, etc.

```
IMAGE          CREATED         CREATED BY                                      SIZE      COMMENT
a8c1ae078d5b   10 seconds ago  /bin/sh -c #(nop)  CMD ["/bin/sh" "-c" "yarn…   0B
9d2d1b910ebc   10 seconds ago  /bin/sh -c apt-get remove -y git python3-pip…   1.91MB
085a5d2da5f4   13 seconds ago  /bin/sh -c tar -xvzf gosec_2.0.0_linux_amd64…   8.85MB
f8290ae97e6a   5 days ago      /bin/sh -c curl -OL https://github.com/secur…   4.72MB
d46051df15a7   5 days ago      /bin/sh -c yarnpkg install                      272MB
bddd59b50ced   5 days ago      /bin/sh -c #(nop) WORKDIR /precaution           0B
396a54161949   5 days ago      /bin/sh -c git clone https://github.com/vmwa…   2.43MB
8ab12298d2da   5 days ago      /bin/sh -c apt-get update && apt-get upgrade…   583MB
85c4fd36a543   6 weeks ago     /bin/sh -c #(nop)  CMD ["bash"]                 0B
<missing>      6 weeks ago     /bin/sh -c #(nop) ADD file:99bf629976cd3d79c…   114MB
```

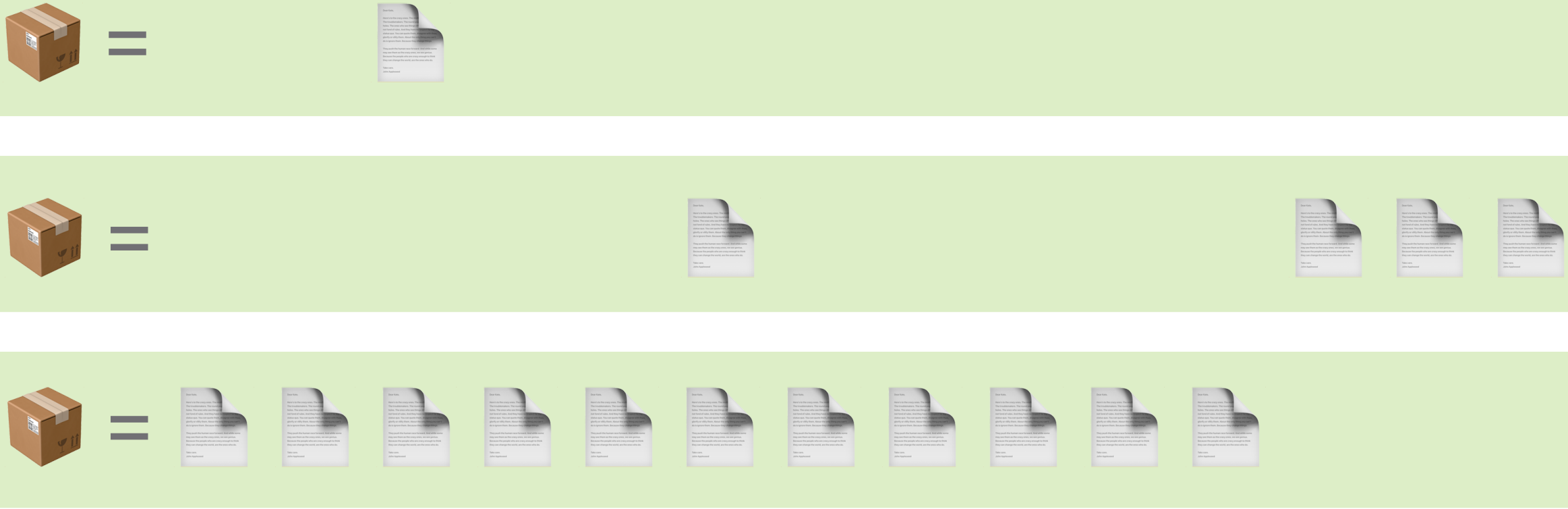
Sidebar: OCIv1 Image Format

Tarballs all the way down



Sidebar: OCIv1 Image Format

Tarballs all the way down



Experiment 2a: packaging Fluentd

Unified logging layer

Why fluentd? (fluentd.org)

- Cloud native application (we all ❤️ logs)
- Ruby - dynamic languages require more diligence

Debian: 😞

- `gem2deb` requires dependencies be satisfiable by apt
  - Ergo; need to package dependencies first

Buildroot: 😫

- No auto-packaging tools exist for buildroot

Yocto Project: 😩

- `recipetool` doesn't support Ruby projects

Guix: 😢

+ `guix import` has a `--recursive` option, to also handle dependencies
- Still couldn't build a working package