Don’t Ship That Container

on the challenges of compliance of container images

Dirk Hohndel
Chief Open Source Officer
Vmware, Inc
April 2018
Containers are easy

Compliance is hard

Let’s go shopping

(apparently not © ca 1992, Mattel)
Assumptions made

... in other words: if you don’t like this talk, it’s your own fault

- Audience understands open source license compliance
- Audience has basic understanding of container technology
- Examples are written in the context of Docker containers
  - But this really isn’t specific to Docker
- Goal is to ship a container image, not a Dockerfile so the user can build their own image (as a workaround for “distribution”)
A Super Simple Example

... setting the stage

```bash
hohndel@ubuntu:~$ cat Dockerfile
FROM debian
CMD [ "/bin/echo", "Hello LLW2018" ]
```

```bash
hohndel@ubuntu:~$ sudo docker run -it test
Hello LLW2018
```

```bash
hohndel@ubuntu:~$ sudo apt-get dist-upgrade
```

81 packages installed

To upgrade to a new release:

Once you have performed the above steps to switch to the old-releases branch, close the Update Manager and then do:

```
sudo apt-get update
dsudo apt-get install update-manager
```

See also EOLUpgrades » Community Help Wiki.

Other contributors (alphabetically):

Andrew G. Betts, Timmiusz Brzezinski, Jan Fabian, M.K., Chas..., Mark Cutajar, Miguel del Olmo, Dario...
What Could Possibly Go Wrong?

... Dockerfiles are never as simple as the simple examples make you believe

```bash
FROM debian:jessie
RUN set -ex; \\
    wget -O /usr/local/bin/gosu \ 
    "https://github.com/tianon/gosu/releases/download/1.10/gosu-amd64" ; \\
    chmod +x /usr/local/bin/gosu
```
What Could Possibly Go Wrong?

... people do incredibly dumb stuff

```bash
RUN echo "deb https://repo.NOPE.com/apt jessie main" > \
    /etc/apt/sources.list.d/nope.list \
    && { \
        echo 'Package: *'; \
        echo 'Pin: release o=Our Dev Team'; \
        echo 'Pin-Priority: 998'; \
    } > /etc/apt/preferences.d/nope ; \
apt-get update && update upgrade -y
```
Even standard practices raise questions

... there are no simple cases here

```
FROM debian:stable  # could be different next time you run it
RUN apt-get update  # will likely change almost every time you run it
RUN apt-get install -y some-app  # from upstream repo, but also point in time
COPY docker-entrypoint /usr/bin/  # from local file system – track sources
ENTRYPOINT ["/usr/bin/docker-entrypoint"]
```
It Gets Worse

Most people just start with a Dockerfile they “find somewhere”

Let’s look at “elasticsearch”

Which uses openjdk:8-jre
It Gets Worse

Most people just start with a Dockerfile they “find somewhere”

Let’s look at “elasticsearch”

Which uses openjdk:8-jre

Which uses buildpack-deps:stretch-curl
It Gets Worse

Most people just start with a Dockerfile they “find somewhere”

Let’s look at “elasticsearch”

Which uses openjdk:8-jre

Which uses buildpack-deps:stretch-curl

Which uses debian:stretch
It Gets Worse

Recursive challenge of finding the dependency tree of Dockerfiles
Determining the licenses and corresponding sources for each of the components
At the right point in time
This is **HARD** if done at build time.
This is **PRETTY MUCH IMPOSSIBLE** after the fact

What are the implications for that nice Container image you want to ship?
Summary

... “what are you complaining about?”

- Which packages are included?
- In which version / which patches applied?
- Under which licenses?
- Where are the corresponding sources?
- Is this stable across re-builds?

The tooling is designed to make it quick and easy to throw together an image.

The tooling (unintentionally?) makes it very easy to create compliance nightmares.
Solutions
... “what should we do?”

- DON’T SHIP THAT CONTAINER!
- ...or...
  - Start with a known-good (i.e. OSL / corresponding sources available), versioned base
  - Avoid anti-patterns of installing random <censored>
  - Train engineers and compliance teams about the pitfalls of container build systems
Solutions

... “what should we do?”

- Track OSL / corresponding source for your layers
- Use sane SCM practices to track this over time
- Make that information available to users
- Use tooling to match your layers with meta information about content
  - Simplest form: annotate and publish Dockerfiles
  - Attempt to automate (WIP): https://github.com/vmware/tern
Tern – Starting a Community Project for Content Discovery

... identify your container’s bill of materials

FROM vmware/photon:1.0

RUN yum install -y git && yum clean all

RUN commands

Base Image
Q&A
Thank You