Shipping Compliant Container Images

Nisha Kumar
Dirk Hohndel
VMware, Inc
March 2019
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

```
hohndel@ubuntu:~/docker$ cat Dockerfile
FROM debian

CMD [ "/bin/echo", "Hello Compliance Experts!""]

hohndel@ubuntu:~/docker$ sudo docker build -t test.
Sending build context to Docker daemon 2.048kB
Step 1/2 : FROM debian
latest: Pulling from library/debian
05d1a5232b46: Pull complete
Digest: sha256:07fe888a6090482fc6e930c1282d1edf67998a39a09a0b339242fbfa2b602fff
Status: Downloaded newer image for debian:latest
  ---> f2aae6ff5d89
Step 2/2 : CMD [ "/bin/echo", "Hello Compliance Experts!""]
  ---> Running in 5e11bdcd2c99
Removing intermediate container 5e11bdcd2c99
  ---> 829ca494d0fe
Successfully built 829ca494d0fe
Successfully tagged test:latest

hohndel@ubuntu:~/docker$ sudo docker run -it test
Hello Compliance Experts!

hohndel@ubuntu:~/docker$
```
A Super Simple Example

... that’s a cute little ”Hello World” program you got there... how big is that?

```bash
hohndel@ubuntu:~/docker$ sudo docker image inspect test --format='{{.Size}}'
100576015
hohndel@ubuntu:~/docker$ echo 100576015 | awk '{ printf "%1fMB\n", $1/1024/1024 }'
95.9MB
```
A Super Simple Example

… OOPS … I wonder what’s inside…
What Could Possibly Go Wrong?

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

```
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

```
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?
Checking your container before shipping your container

A possible way out
Containerizing Apps
Using Docker and Dockerfiles

FROM <image>

RUN <script>

COPY . .

Diff files:
install app dependencies and extra modifications

Diff Files:
Copy app into container

Diff Files: etc/ca-certificates/* usr/share/ca-certificates/*
OS & language package manager dependencies

BaseOS: bin boot etc home lib opt root tmp usr var
Debian, Photon, Alpine

Your Changes

Somebody else's Image

©2019 VMware, Inc.
Check FROM

```
[nisha@localhost tern]$ docker history docker.io/golang

<table>
<thead>
<tr>
<th>IMAGE</th>
<th>CREATED</th>
<th>CREATED BY</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>fb7a47d8605b</td>
<td>4 weeks ago</td>
<td>/bin/sh -c #(nop) WORKDIR /go</td>
<td>0 B</td>
</tr>
<tr>
<td>&lt;missing&gt;</td>
<td>4 weeks ago</td>
<td>/bin/sh -c mkdir -p &quot;$GOPATH/src&quot; &quot;$GOPATH...</td>
<td>0 B</td>
</tr>
<tr>
<td>&lt;missing&gt;</td>
<td>4 weeks ago</td>
<td>/bin/sh -c #(nop) ENV PATH=/go/bin:/usr/...</td>
<td>0 B</td>
</tr>
<tr>
<td>&lt;missing&gt;</td>
<td>4 weeks ago</td>
<td>/bin/sh -c #(nop) ENV GOPATH=/go</td>
<td>0 B</td>
</tr>
<tr>
<td>&lt;missing&gt;</td>
<td>4 weeks ago</td>
<td>/bin/sh -c set -eux; dpkgArch=&quot;$(dpkg --...</td>
<td>341 MB</td>
</tr>
<tr>
<td>&lt;missing&gt;</td>
<td>4 weeks ago</td>
<td>/bin/sh -c #(nop) ENV GOLANG_VERSION=1.11</td>
<td>0 B</td>
</tr>
<tr>
<td>&lt;missing&gt;</td>
<td>4 weeks ago</td>
<td>/bin/sh -c apt-get update &amp;&amp; apt-get insta...</td>
<td>162 MB</td>
</tr>
<tr>
<td>&lt;missing&gt;</td>
<td>5 weeks ago</td>
<td>/bin/sh -c apt-get update &amp;&amp; apt-get insta...</td>
<td>142 MB</td>
</tr>
<tr>
<td>&lt;missing&gt;</td>
<td>5 weeks ago</td>
<td>/bin/sh -c set -ex; if ! command -v gpg &gt;...</td>
<td>7.8 MB</td>
</tr>
<tr>
<td>&lt;missing&gt;</td>
<td>5 weeks ago</td>
<td>/bin/sh -c apt-get update &amp;&amp; apt-get insta...</td>
<td>23.2 MB</td>
</tr>
<tr>
<td>&lt;missing&gt;</td>
<td>5 weeks ago</td>
<td>/bin/sh -c #(nop) CMD [&quot;bash&quot;]</td>
<td>0 B</td>
</tr>
<tr>
<td>&lt;missing&gt;</td>
<td>5 weeks ago</td>
<td>/bin/sh -c #(nop) ADD file:58d5c21fcabcf1e...</td>
<td>101 MB</td>
</tr>
</tbody>
</table>
```
Check RUN

docker history --no-trunc docker.io/golang

```bash
<missing>
5 weeks ago /bin/sh -c set -eux; dpkgArch="$(dpkg --print-architecture)"; case "${dpkgArch#*-*}" in amd64) goRelArch='linux-amd64'; goRelSha256='b3fcf280ff86558e0559e185b601c9eade0fd24c900b4c63cd14d1d38613e499' ;; armhf) goRelArch='linux-armv6l'; goRelSha256='8ffeb3577d8ca5477064f1c8b7839835973c866487f2bf81df1227eaa96826ac';; arm64) goRelArch='linux-arm64'; goRelSha256='e4853168f41d0bea65e4d38f992a2d44b5855260f5623640c5ead89d515c56c9' ;; i386) goRelArch='linux-i386'; goRelSha256='1a91932b65b4af2f84ef2dce10d790e6a0d3d22c9ea1bdf3d8c4d9279d9f680e' ;; ppc64le) goRelArch='linux-ppc64le'; goRelSha256='e874d617f0e322f8c2dda8c23ea3a2ea21d5e7177abb1f8b6a0ac7c6d53272' ;; s390x) goRelArch='linux-s390x'; goRelSha256='c113495fbb175d6beb1b881750de1dd034c7ae8657c30b3def88080329af0a15';) goRelArch='src'; goRelSha256='afc1e12f5fe49a471e3aae7d906c73e9d5b1fdd36d52d72652ddee8f6250152fb'; echo >&2; echo >&2 "warning: current architecture (${dpkgArch}) does not have a corresponding Go binary release; will be building from source"; echo >&2 ;; esac; url="https://golang.org/dl/go$GOLANG_VERSION"; goRelArch}.tar.gz"; wget -O go.tar.gz "$url"; echo "$goRelSha256" | sha256sum -c -; tar -C /usr/local -xzf go.tar.gz; rm go.tar.gz; if [ "$goRelArch" = 'src' ]; then echo >&2; echo >&2 'error: UNIMPLEMENTED'; echo >&2 'TODO install golang-any from jessie-backports for GOROOT_BOOTSTRAP (and uninstall after build)'; echo >&2: exit 1; fi; export PATH="/usr/local/go/bin:$PATH": "$go version 341 MB
```
Check the whole container image

```
[nisha@localhost ~]$ mkdir image
[nisha@localhost ~]$ time docker save docker.io/golang | tar xC image/.
real  0m19.899s
user  0m0.326s
sys   0m1.788s
[nisha@localhost ~]$ cd image/
[nisha@localhost image]$ ls -al
total 52
drwxrwxr-x.  9 nisha nisha 4096 Oct 12 17:49 .
- rw-r--r--. 25 nisha nisha 4096 Oct 12 17:48 ..
drwxr-xr-x.  2 nisha nisha 4096 Sep  5 04:06 1eff74c9194e6e74bea2cfffced0eaf66367daff22f97b4cfd489012aaddcfd2
-drwxr-xr-x.  2 nisha nisha 4096 Sep  5 04:06 2df07e043bcbbcf7f629b6ea2990401333218998c0b9fb8eb02621bd46f35
-drwxr-xr-x.  2 nisha nisha 4096 Sep  5 04:06 6e1bf8ee3b708647f06cac5e20e6998c1f55d1e89b311065c82727c7abbde134
-drwxr-xr-x.  2 nisha nisha 4096 Sep  5 04:06 79babc7f8c8b23ec155b78f7462db48b7ab383763af26f40ce8ba650c6d75f6
-drwxr-xr-x.  2 nisha nisha 4096 Sep  5 04:06 89f2388507f97645bcd99e616e131eaac0188e37ec495d8cb56ad60f9e8df0d
-drwxr-xr-x.  2 nisha nisha 4096 Sep  5 04:06 c96b61548d40e795197b8dd61fb531ab25ef51b592548a2b7baca71e5fe4c8c
-drwxr-xr-x.  2 nisha nisha 4096 Sep  5 04:06 ce9faa005e4d6e0166d521b875722528155f6d923d1482cedb2aa67670b33d9e
-rw-r--r--.  1 nisha nisha 5456 Sep  5 04:06 fb7a47d8605b86174e88a730064bb877a7d100ac31df3e46cc8a829160d62136
json
-rw-r--r--.  1 nisha nisha 674 Dec 31 1969 manifest.json
-rw-r--r--.  1 nisha nisha 99 Dec 31 1969 repositories
```

Container images have files and directories
One method - file scanning

Open Source
- Clair (de facto for container scanning)
- Scancode
- FOSSology
- Anchore (engine)

Commercial
- Twistlock
- Docker Security
- AquaSec
- Blackduck
- FOSSA
Another method - use "containers"

- `sudo mount -t proc /proc /path/to/rootfs/proc`
- `sudo mount -o bind /sys /path/to/rootfs/sys`
- `sudo mount -o bind /dev /path/to/rootfs/dev`
- `sudo cp /etc/resolv.conf /path/to/rootfs/etc/resolv.conf`
- `sudo chroot rootfs /bin/bash -c "dpkg --get-selections"`
Tern Automates Compliance for Containers
Recorded demo
Tern’s results
Features

- Support for Debian, Ubuntu, Photon and Alpine package managers
- Lists Packages installed and their Dependencies
- Extensible Architecture (add your own method of license and source information)
- Caching by Container Image Layer
- Can be used as a standalone tool to help container developers or part of a container build and release pipeline
- Structured data output (JSON and YAML)
- Active community
Future work

- SPDX document support
- Enable language package managers
- Enable external files
  - GitHub repositories
  - SPDX license identifiers
  - Call out to external tools
- Custom packages
  - Hardcoded values
Thank You