

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?

```
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...

```
hohndel@ubuntu:~/docker$ sudo docker container run -it test /bin/bash
root@56c40e2f4ddb:/# dpkg --get-selections | grep -v deinstall | cut -f 1 | pr -T -4
adduser          init-system-helpe libgpg-error0:amd libtinfo5:amd64
apt              iproute2          libidn11:amd64    libudev1:amd64
base-files       iputils-ping       liblz4-1:amd64    libustr-1.0-1:amd
base-passwd      libacl1:amd64      liblzma5:amd64    libuuid1:amd64
bash             libapt-pkg5.0:amd libmnl0:amd64     login
bsdutils         libattr1:amd64     libmount1:amd64   lsb-base
coreutils        libaudit-common    libncursesw5:amd6 mawk
dash             libaudit1:amd64    libnettle6:amd64  mount
debconf          libblkid1:amd64    libpam-modules:am multiarch-support
debian-archive-ke libbz2-1.0:amd64   libpam-modules-bi ncurses-base
debianutils      libc-bin           libpam-runtime     ncurses-bin
diffutils        libc6:amd64         libpam0g:amd64     passwd
dpkg             libcap-ng0:amd64   libpcre3:amd64     perl-base
e2fslibs:amd64   libcap2:amd64      libselinux1:amd64 sed
e2fsprogs        libcomerr2:amd64   libsemanage-commo sensible-utils
findutils        libdb5.3:amd64     libsemanage1:amd6 sysvinit-utils
gcc-6-base:amd64 libdebconfclient0 libsepol1:amd64    tar
gpgv            libelf1:amd64      libsmartcols1:amd tzdata
grep            libfdisk1:amd64    libss2:amd64       util-linux
gzip            libgcc1:amd64      libstdc++6:amd64   zlib1g:amd64
hostname         libgcrypt20:amd64  libsystemd0:amd64
root@56c40e2f4ddb:/# dpkg --get-selections | grep -v deinstall | cut -f 1 | wc -l
83
```

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

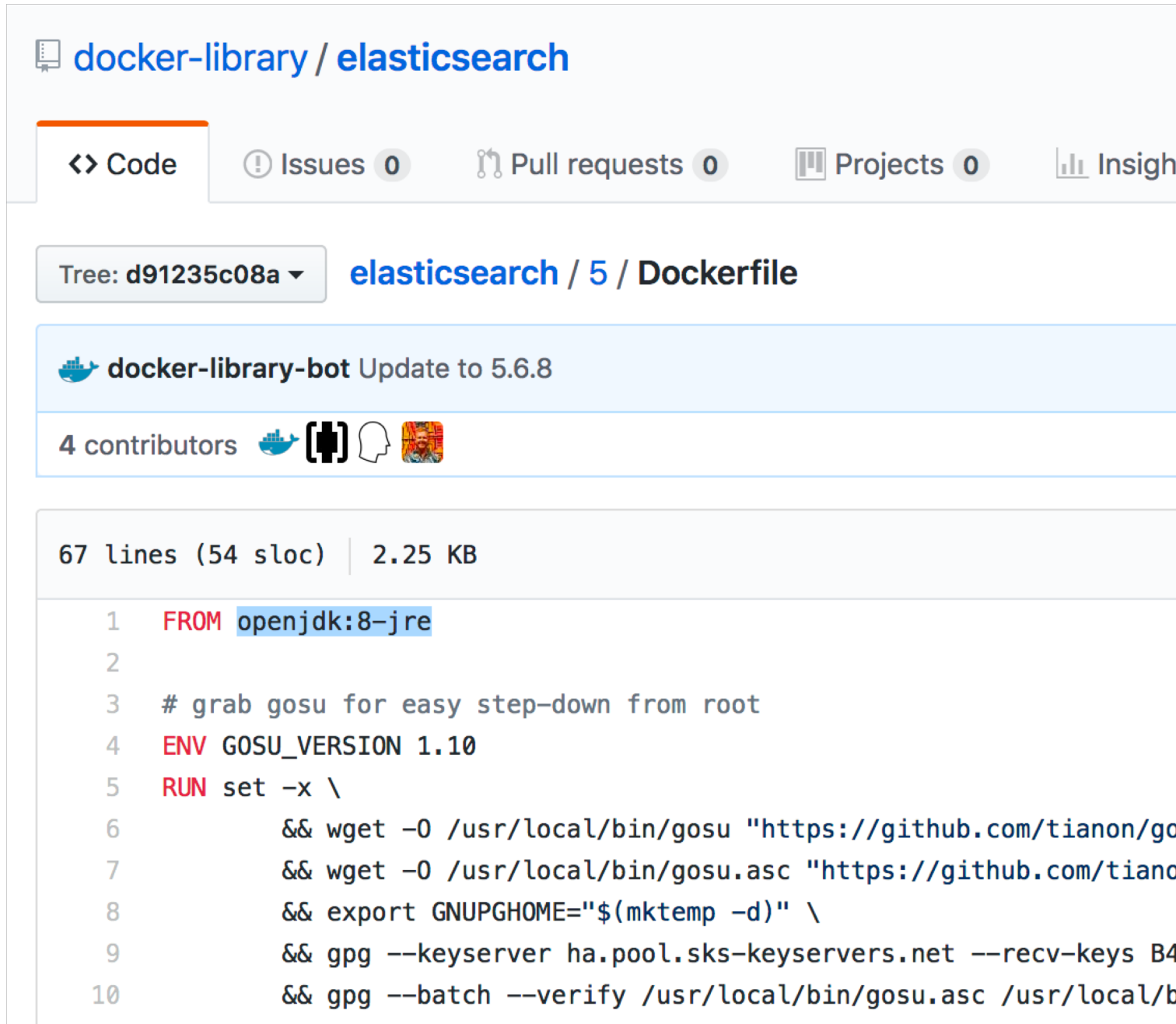
```
FROM debian:stable          # could be different next time you run it
RUN apt-get update          # will likely change almost every time you run it
RUN apt-get install -y some-app  # from upstream repo, but also point in time
COPY docker-entrypoint /usr/bin/  # from local file system - track sources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



The screenshot shows the Docker Hub interface for the `docker-library / elasticsearch` repository. The `Code` tab is selected, displaying the Dockerfile for the `elasticsearch / 5 / Dockerfile` image. The Dockerfile content is as follows:

```
1 FROM openjdk:8-jre
2
3 # grab gosu for easy step-down from root
4 ENV GOSU_VERSION 1.10
5 RUN set -x \
6     && wget -O /usr/local/bin/gosu "https://github.com/tianon/gosu"
7     && wget -O /usr/local/bin/gosu.asc "https://github.com/tianon/gosu"
8     && export GNUPGHOME="$(mktemp -d)" \
9     && gpg --keyserver ha.pool.sks-keyservers.net --recv-keys B4
10    && gpg --batch --verify /usr/local/bin/gosu.asc /usr/local/b
```

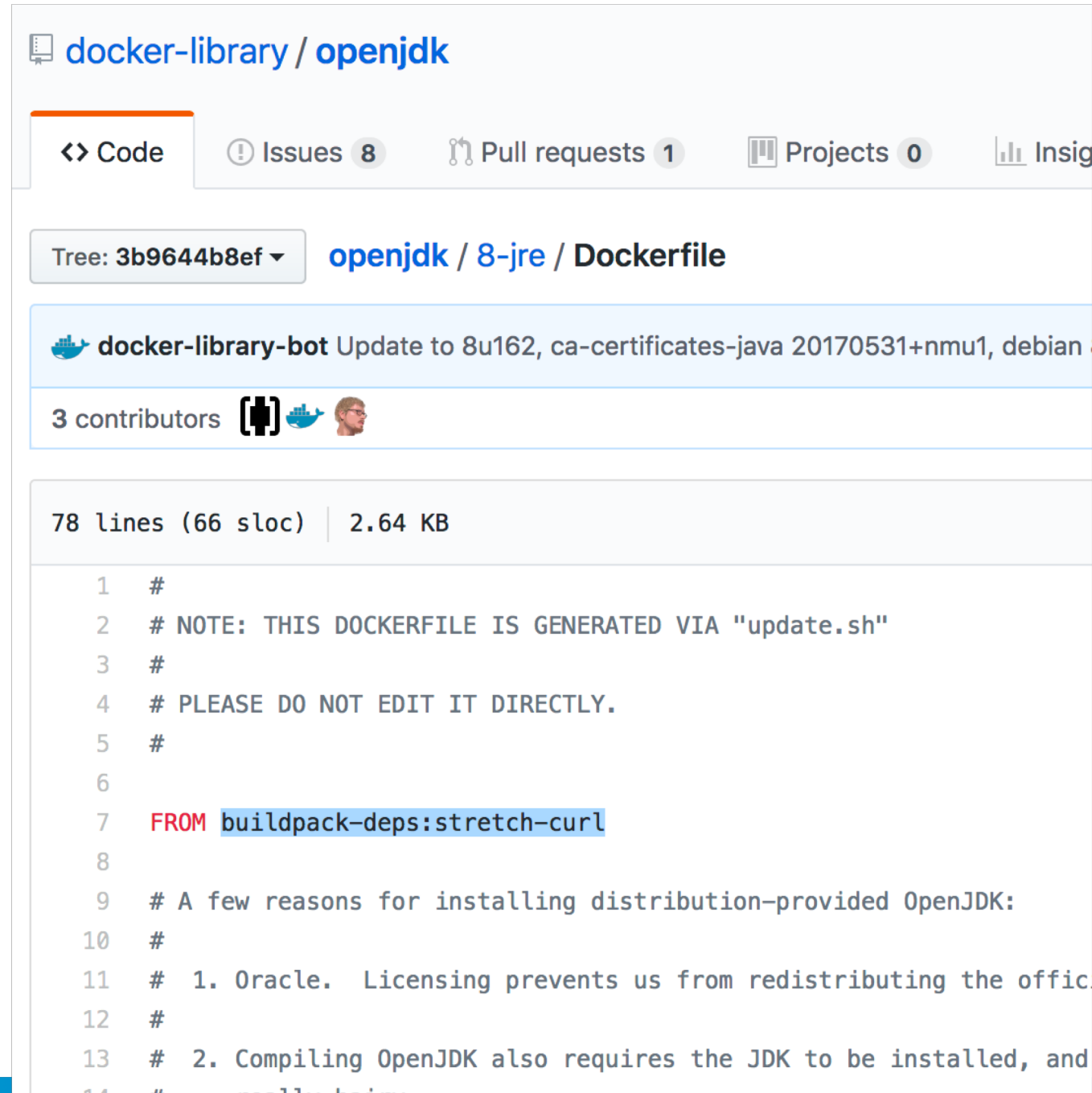
It Gets Worse

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`



The screenshot shows the Docker Hub interface for the `openjdk` repository. The breadcrumb navigation shows `openjdk / 8-jre / Dockerfile`. A commit message from `docker-library-bot` is visible: "Update to 8u162, ca-certificates-java 20170531+nmu1, debian". Below this, it says "3 contributors". The file details show "78 lines (66 sloc) | 2.64 KB". The Dockerfile content is displayed with line numbers 1 through 14. Line 7 contains the instruction `FROM buildpack-deps:stretch-curl`, which is highlighted with a blue selection box. The code is as follows:

```
1  #
2  # NOTE: THIS DOCKERFILE IS GENERATED VIA "update.sh"
3  #
4  # PLEASE DO NOT EDIT IT DIRECTLY.
5  #
6
7  FROM buildpack-deps:stretch-curl
8
9  # A few reasons for installing distribution-provided OpenJDK:
10 #
11 # 1. Oracle. Licensing prevents us from redistributing the offic
12 #
13 # 2. Compiling OpenJDK also requires the JDK to be installed, and
14 #
```

It Gets Worse

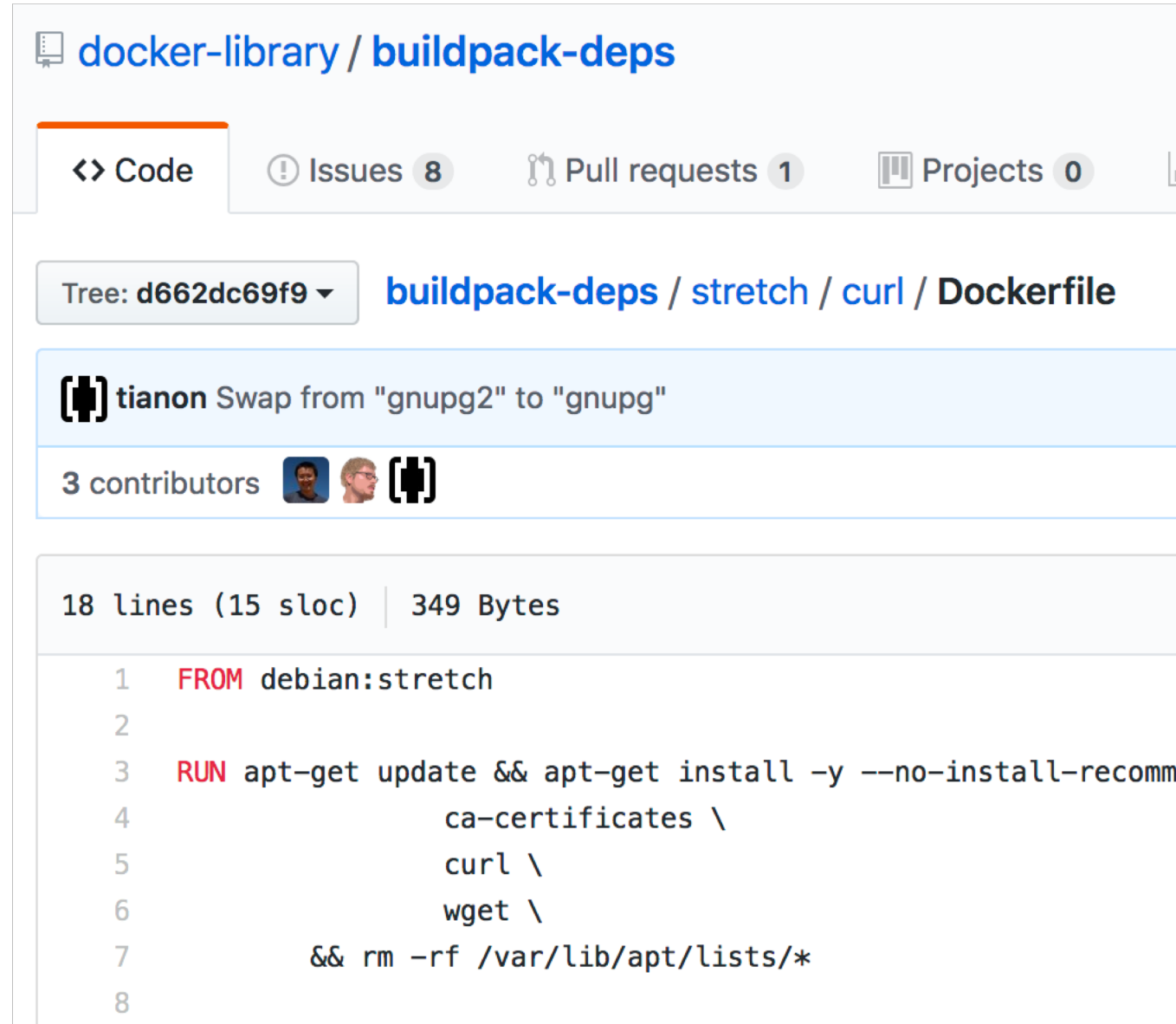
Most people just start with a Dockerfile they “find somewhere”

Let’s look at “elasticsearch”

Which uses `openjdk:8-jre`

Which uses `buildpack-deps:stretch-curl`

Which uses `debian:stretch`



The screenshot shows the Docker Hub interface for the `buildpack-deps` repository. The breadcrumb navigation indicates the path: `docker-library / buildpack-deps`. Below the navigation bar, there are tabs for `Code`, `Issues` (8), `Pull requests` (1), and `Projects` (0). The current view is the `Dockerfile` for the `stretch / curl` image. The commit hash is `d662dc69f9`. A commit message by `tianon` is visible: "Swap from 'gnupg2' to 'gnupg'". Below this, it shows 3 contributors. The Dockerfile details show 18 lines (15 sloc) and 349 Bytes. The Dockerfile content is as follows:

```
1 FROM debian:stretch
2
3 RUN apt-get update && apt-get install -y --no-install-recomm
4     ca-certificates \
5     curl \
6     wget \
7     && rm -rf /var/lib/apt/lists/*
8
```


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

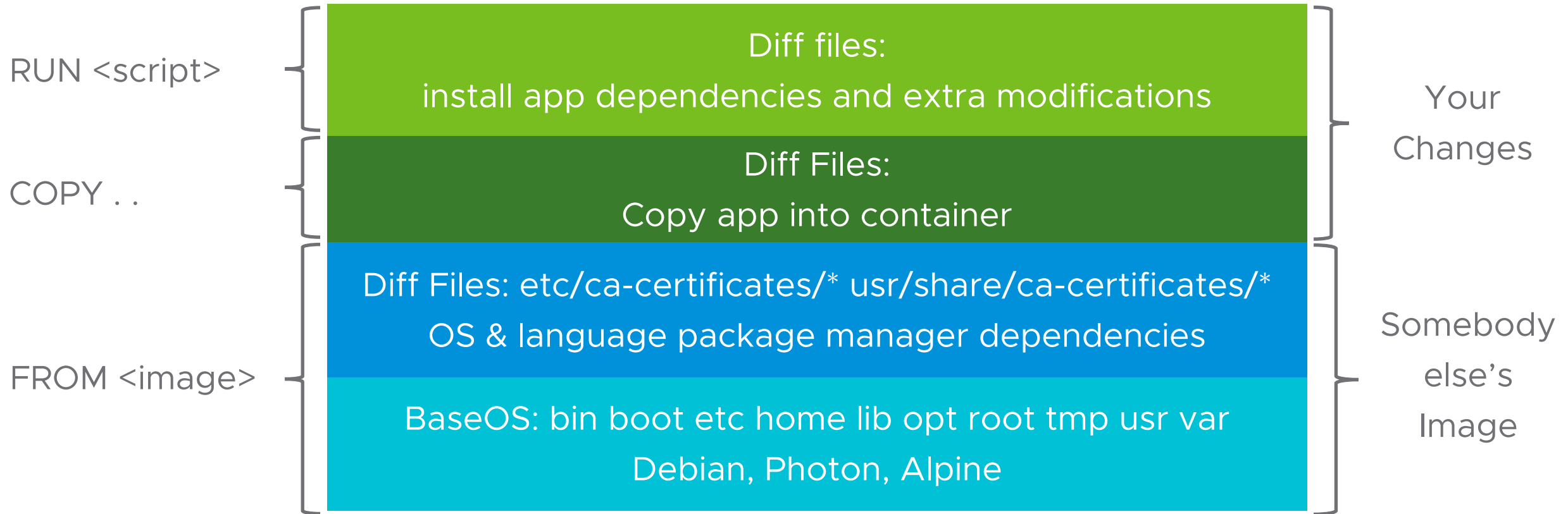
What are the implications for that nice Container image you want to ship?

Checking your container before shipping your container

A possible way out

Containerizing Apps

Using Docker and Dockerfiles



Check FROM

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

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

Check RUN

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

```
<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='8ffeb3577d8ca5477064f1cb8739835973c866487f2bf81df1227eaa96826acd' ;; arm64) goRelArch
='linux-arm64'; goRelSha256='e4853168f41d0bea65e4d38f992a2d44b58552605f623640c5ead89d515c56c9' ;; i386) goR
elArch='linux-386'; goRelSha256='1a91932b65b4af2f84ef2dce10d790e6a0d3d22c9ea1bdf3d8c4d9279dfa680e' ;; ppc64
el) goRelArch='linux-ppc64le'; goRelSha256='e874d617f0e322f8c2dda8c23ea3a2ea21d5dfe7177abb1f8b6a0ac7cd653272'
;; s390x) goRelArch='linux-s390x'; goRelSha256='c113495fbb175d6beb1b881750de1dd034c7ae8657c30b3de8808032c9
af0a15' ;; *) goRelArch='src'; goRelSha256='afc1e12f5fe49a471e3aae7d906c73e9d5b1fdd36d52d72652dde8f6250152f
b'; 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.tgz "$url"; echo "${goRelSha256} *go.tgz" | sha256sum -c -; tar -C /usr/lo
cal -xzf go.tgz; rm go.tgz; 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 .
drwx----- 25 nisha nisha 4096 Oct 12 17:48 ..
drwxr-xr-x.  2 nisha nisha 4096 Sep  5 04:06 1eff74c9194e6e74bea2cffdced0eaf66367daff22f97b4cfd489012aaadcfb2
drwxr-xr-x.  2 nisha nisha 4096 Sep  5 04:06 2fdf07b43bcbbcff7f629b6ea299040133321899b8c0b9fb8eb02621bd4c6f35
drwxr-xr-x.  2 nisha nisha 4096 Sep  5 04:06 6e1bf8ee3b708647f06cac5e20e6998c1f55d1e89b311065c82727c7abbde134
drwxr-xr-x.  2 nisha nisha 4096 Sep  5 04:06 79bbbc7f8cc8b23ec155b78f7462d48b7ab383763af26f40ce8ba650c6d757f6
drwxr-xr-x.  2 nisha nisha 4096 Sep  5 04:06 89f2388507f97645bcd99e616e1313eaac0188e37ec495d8cb56ad06f9e8dfd0
drwxr-xr-x.  2 nisha nisha 4096 Sep  5 04:06 c96b61548d40e795197b8dd61fb531ab250ef51b592548a2b7baca71e5fe4c8c
drwxr-xr-x.  2 nisha nisha 4096 Sep  5 04:06 ce9faa005e4d6e0166d521b875722528155f6d923d1482cedb2aa67670b3dd9e
-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

```
[nisha@localhost image]$ ls 2fdf07b43bcbbcff7f629b6ea299040133321899b8c0b9fb8eb02621bd4c6f35/contents/etc/
```

adduser.conf	fstab	ld.so.cache	os-release	rc5.d	staff-group-for-usr-local
alternatives	gai.conf	ld.so.conf	pam.conf	rc6.d	subgid
apt	group	ld.so.conf.d	pam.d	rcS.d	subuid
bash.bashrc	group-	libaudit.conf	passwd	resolv.conf	systemd
bindresvport.blacklist	gshadow	localtime	passwd-	rmt	terminfo
cron.daily	host.conf	login.defs	profile	securetty	timezone
debconf.conf	hostname	logrotate.d	profile.d	security	update-motd.d
debian_version	init.d	machine-id	rc0.d	selinux	
default	iproute2	mke2fs.conf	rc1.d	shadow	
deluser.conf	issue	motd	rc2.d	shadow-	
dpkg	issue.net	nsswitch.conf	rc3.d	shells	
environment	kernel	opt	rc4.d	skel	

One method - file scanning

Open Source

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

Commercial

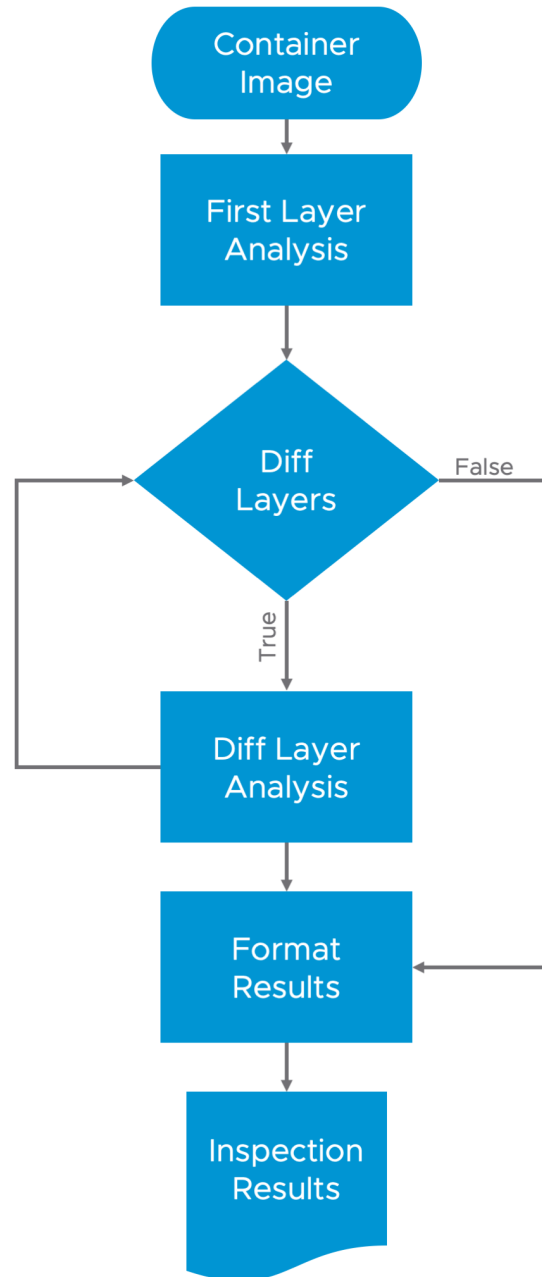
- Twistlock
- Docker Security
- AquaSec
- Blackduck
- FOSSA

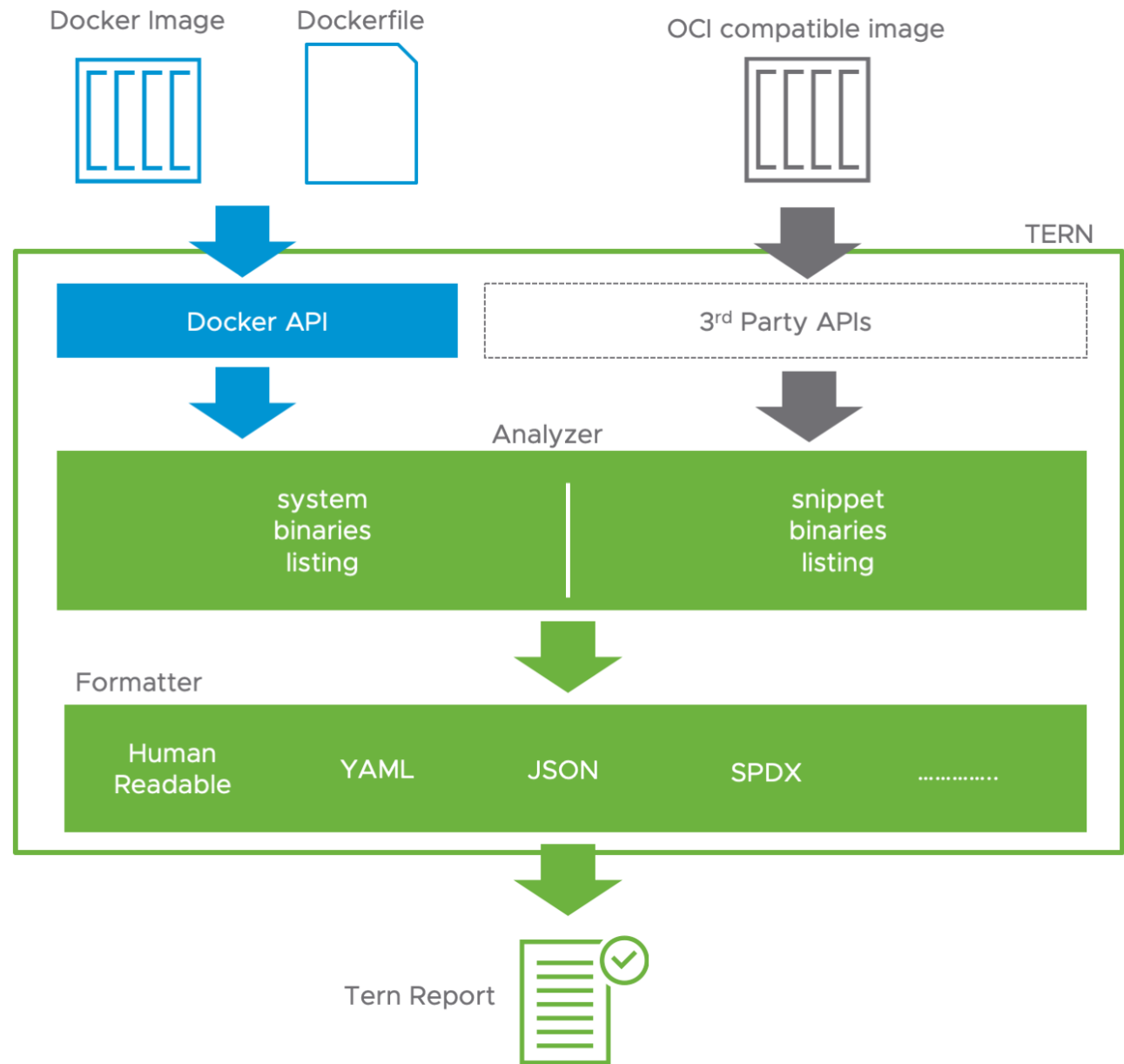
Another method - use "containers"

- `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