My container image has 500 vulnerabilities

Now what ??

Matt Jarvis | Director, Developer Relations | matt.jarvis@snyk.io

$whoami

- Matt Jarvis
  - Director of Developer Relations @ Snyk
- Co-founder Cloud Native Manchester
- Co-founder Cloud Native Edinburgh
- Founder Cloud Natives UK

@mattj_io

mattj-io

mattjarvis.org.uk

purpledobie/mattj-goof:latest

DIGEST: sha256:0dbd9f2811c91b6b2355fae1f82d195b9dc900d79e696fb3dde2595f3660db57

OS/ARCH	COMPRESSED SIZE	LAST PUSHED	VULNERABILITIES
linux/amd64	374 MB	2 minutes ago by mattjarvis	230 H 196 M 473 L Scanned a minute ago

Image Layers | **Vulnerabilities 899**

Vulnerability scanning best practices

- Use **docker scan** to test your local images for vulnerabilities before pushing to Docker Hub
- Connect Docker Hub to Snyk for guidance on remediating vulnerabilities in your image
- Read the Docker documentation to learn more

Fix issues with Snyk

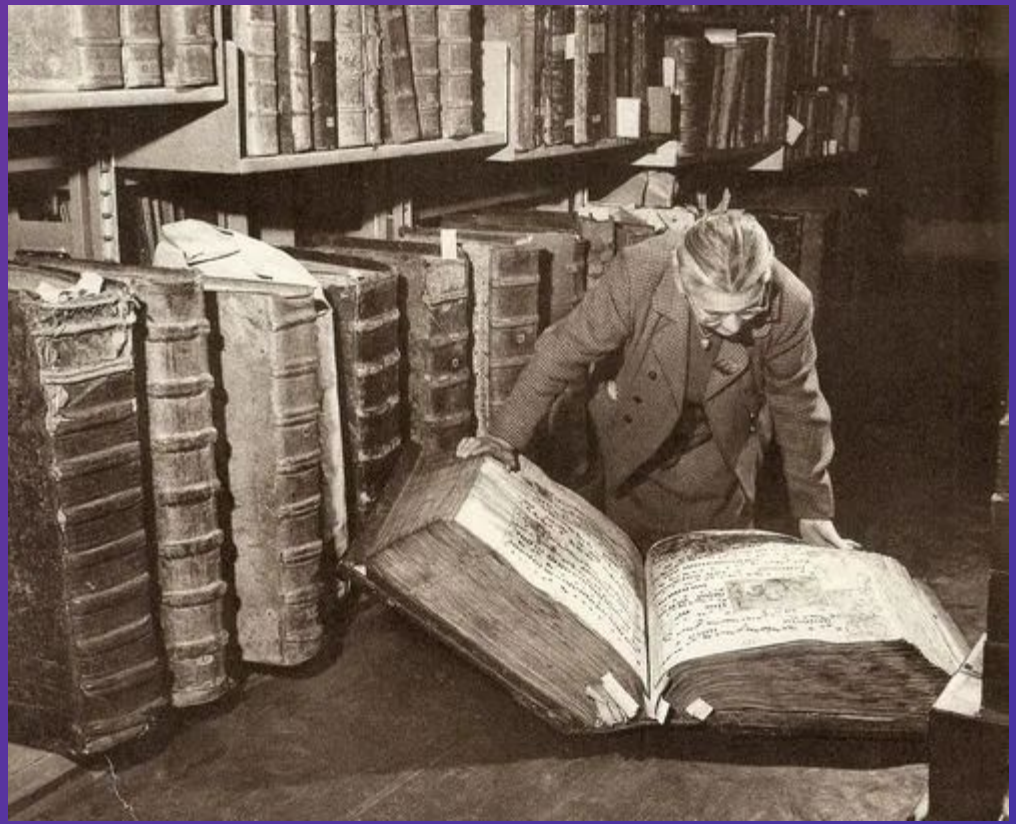
snyk

Severity & Vulnerability		Package	Version	Fixed in
H 9.8 Out-of-bounds Write	CVE-2019-12900	bzip2	1.0.6-8.1	---
H 9.8 Buffer Overflow	CVE-2019-5482	curl	7.52.1-5+deb9u9	7.52.1-5+deb9u10
H 9.8 Double Free	CVE-2019-5481	curl	7.52.1-5+deb9u9	7.52.1-5+deb9u10
H 9.8 Out-of-bounds Write	CVE-2019-18218	file	1:5.30-1+deb9u2	1:5.30-1+deb9u3
H 9.8 CVE-2019-1353	CVE-2019-1353	git	1:2.11.0-3+deb9u4	1:2.11.0-3+deb9u5
H 9.8 NULL Pointer Dereference	CVE-2018-16428	glib2.0	2.50.3-2	2.50.3-2+deb9u1
H 9.8 Race Condition	CVE-2019-12450	glib2.0	2.50.3-2	2.50.3-


OH CRAP!

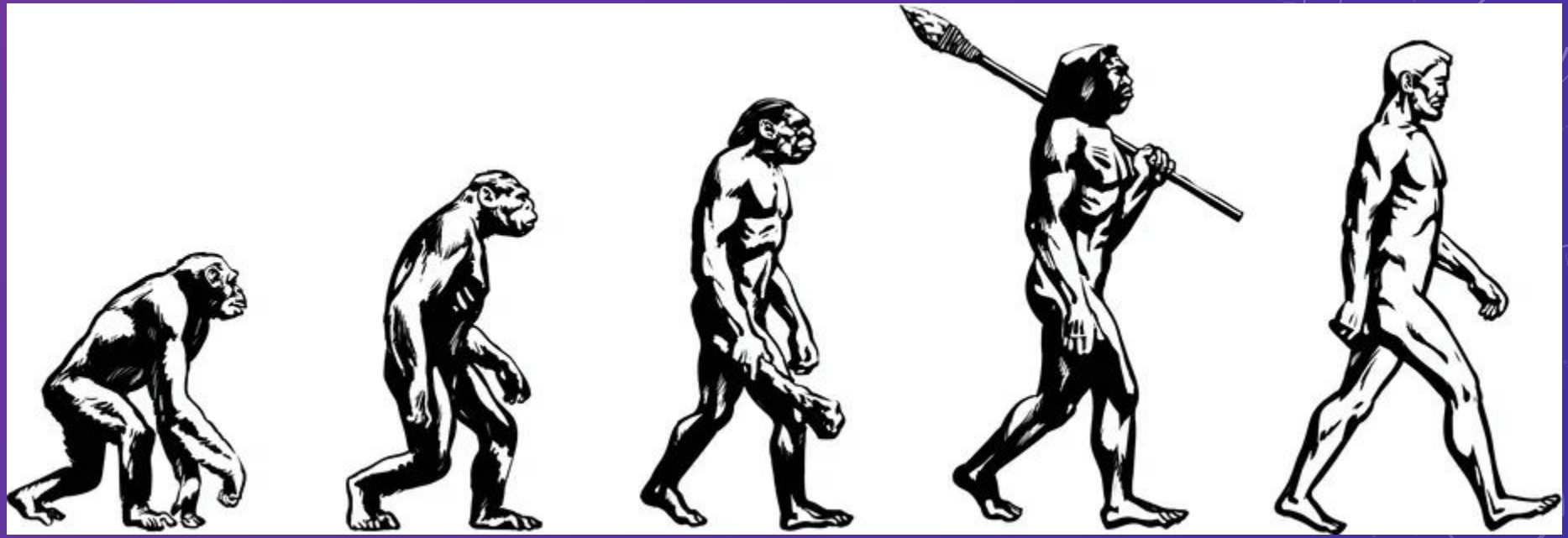
Did I do something ?

Or did I inherit a problem ?

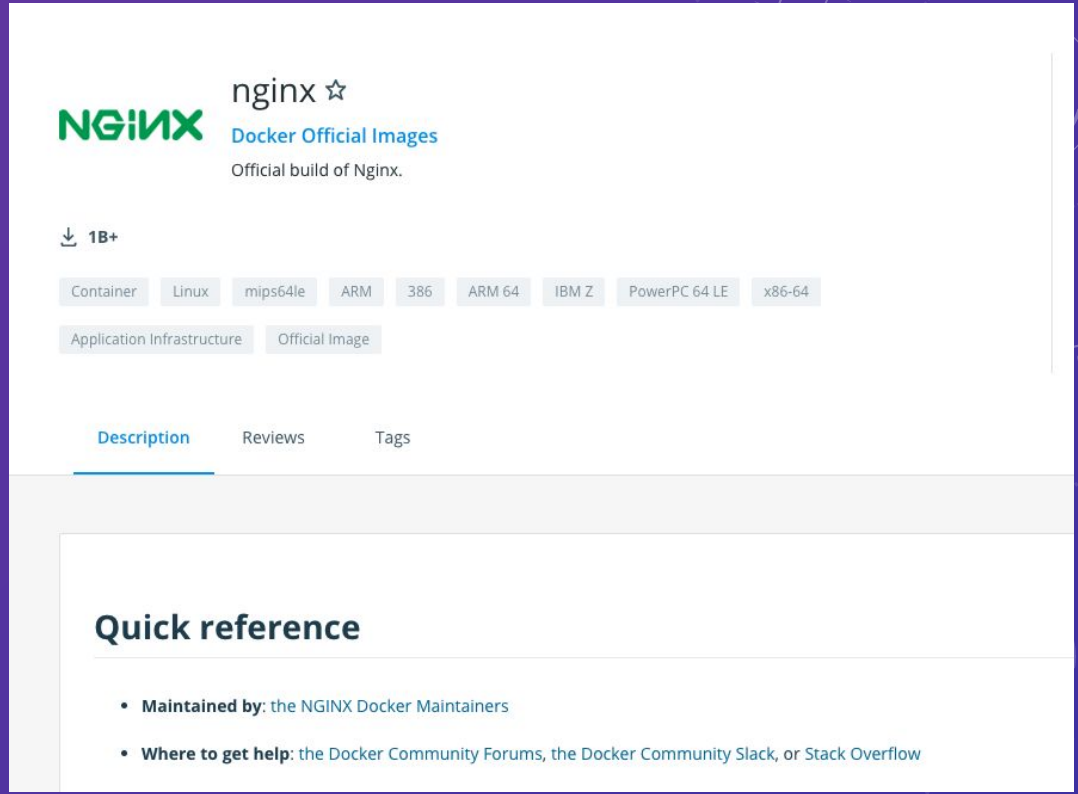



Your code
Your code dependencies
Your added layers
Base Image

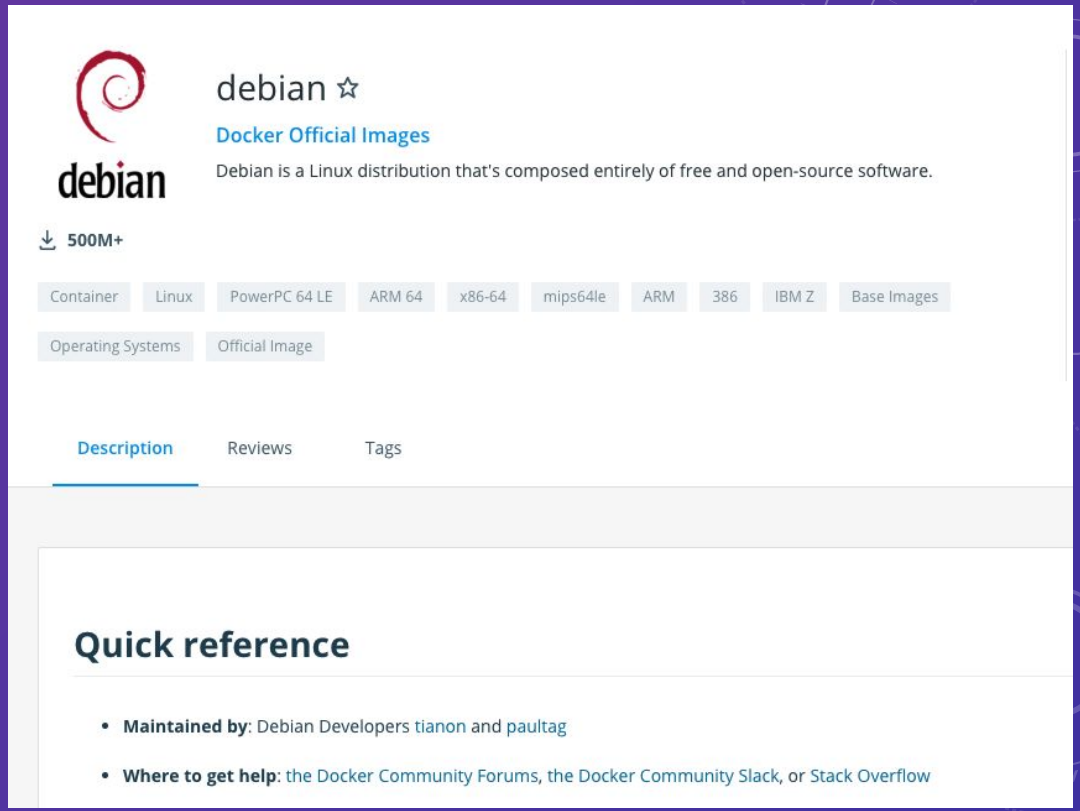

Your code
Your code dependencies
Your added layers
Base Image

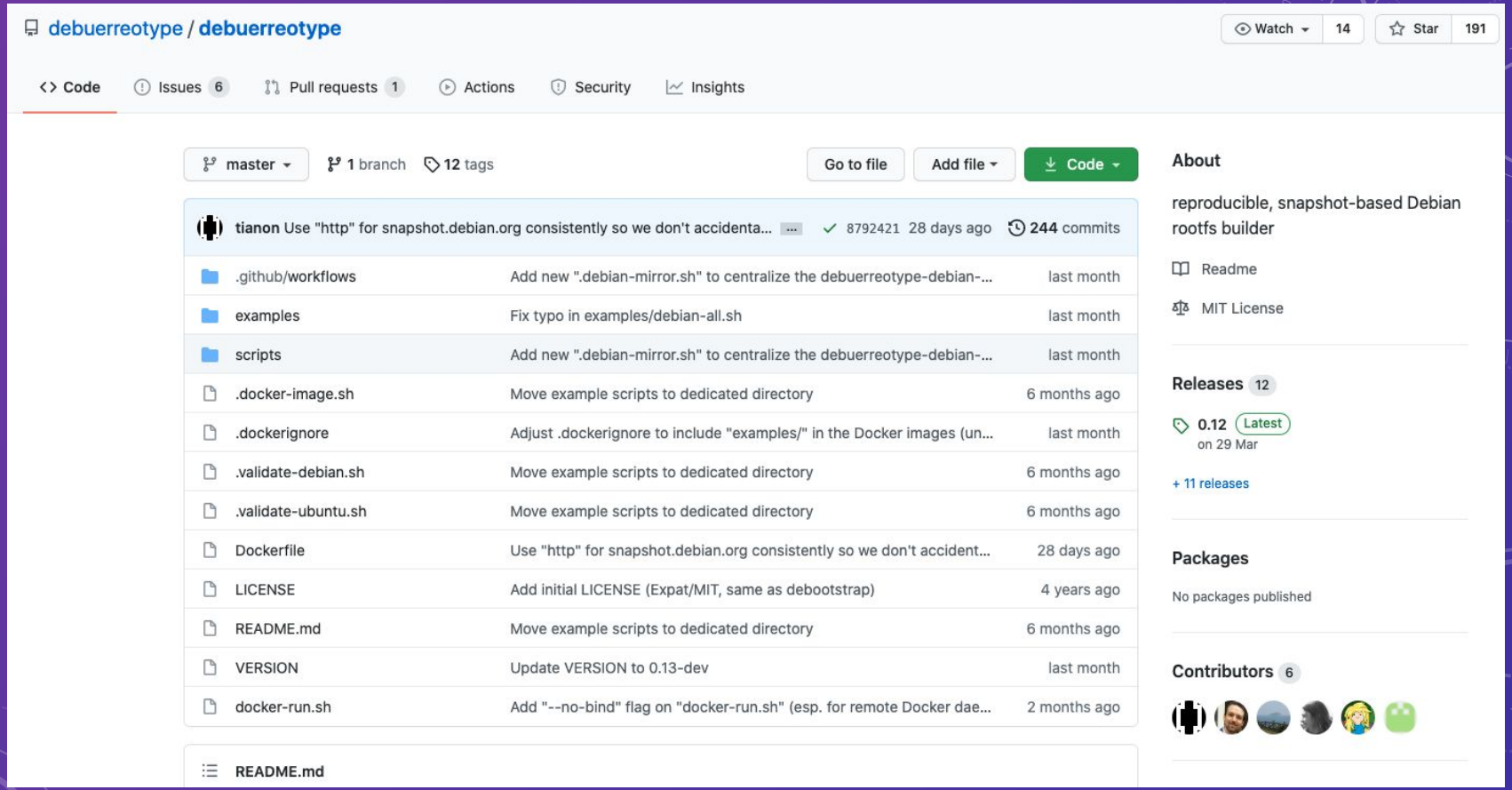


```
FROM debian:buster-slim


LABEL maintainer="NGINX Docker Maintainers
<docker-maint@nginx.com>"


ENV NGINX_VERSION   1.19.7
ENV NJS_VERSION     0.5.1
ENV PKG_RELEASE     1~buster


RUN set -x \
# create nginx user/group first, to be consistent
throughout docker variants
   && addgroup --system --gid 101 nginx \
   && adduser --system --disabled-login --ingroup
nginx --no-create-home --home /nonexistent --gecos
"nginx user" --shell /bin/false --uid 101 nginx \
   && apt-get update \
   && apt-get install --no-install-recommends
--no-install-suggests -y gnupg1 ca-certificates \
***SNIPPED***
```



```
FROM scratch
ADD rootfs.tar.xz /
CMD ["bash"]
```




debuerreotype / debuerreotype
Watch 14
Star 191
Code
Issues 6
Pull requests 1
Actions
Security
Insights
master
1 branch
12 tags
Go to file
Add file
Code
tianon Use "http" for snapshot.debian.org consistently so we don't accidenta... 8792421 28 days ago 244 commits
.github/workflows Add new ".debian-mirror.sh" to centralize the debuerreotype-debian-... last month
examples Fix typo in examples/debian-all.sh last month
scripts Add new ".debian-mirror.sh" to centralize the debuerreotype-debian-... last month
.docker-image.sh Move example scripts to dedicated directory 6 months ago
.dockerignore Adjust .dockerignore to include "examples/" in the Docker images (un... last month
.validate-debian.sh Move example scripts to dedicated directory 6 months ago
.validate-ubuntu.sh Move example scripts to dedicated directory 6 months ago
Dockerfile Use "http" for snapshot.debian.org consistently so we don't accident... 28 days ago
LICENSE Add initial LICENSE (Expat/MIT, same as debootstrap) 4 years ago
README.md Move example scripts to dedicated directory 6 months ago
VERSION Update VERSION to 0.13-dev last month
docker-run.sh Add "--no-bind" flag on "docker-run.sh" (esp. for remote Docker dae... 2 months ago
README.md
About
reproducible, snapshot-based Debian rootfs builder
Readme
MIT License
Releases 12
0.12 Latest
on 29 Mar
+ 11 releases
Packages
No packages published
Contributors 6

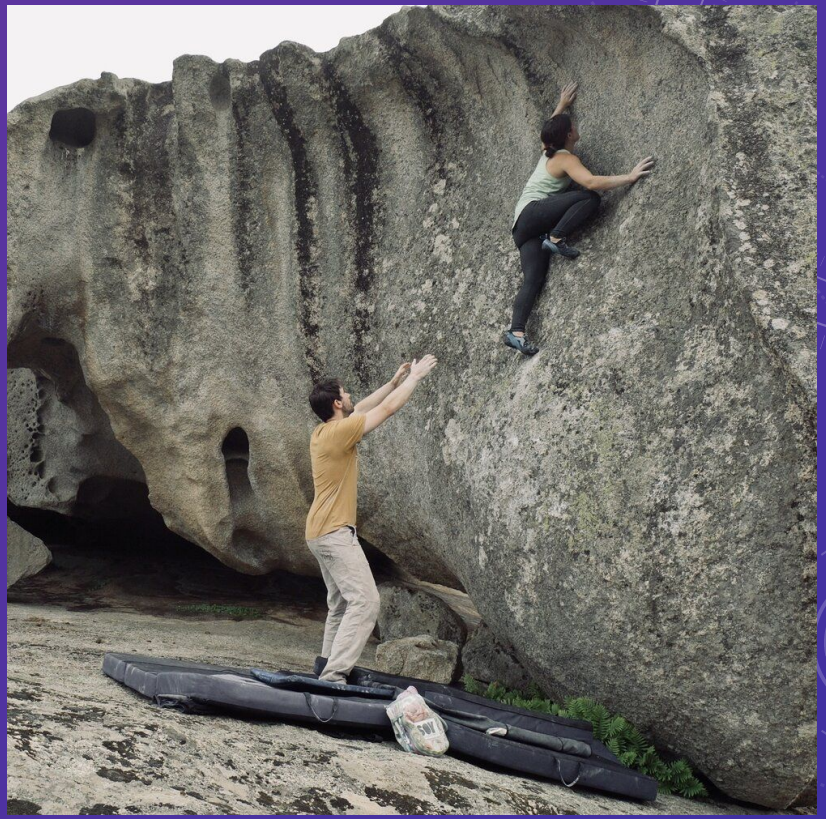
```
FROM scratch
COPY hello /
CMD ["/hello"]
```

Scratch or Upstream ?

Don’t try to solve upstream problems downstream

Trust Factors

Is it maintained ?
Does it have broad adoption ?
Who’s supporting it ?



PICKING A BASE IMAGE ISN'T AS SIMPLE AS IT SEEMS ...

ruby
- 850 MB
- 414 dependencies
- 257 vulnerabilities
- **35 high severity**

ruby:3-slim
- 157 MB
- 107 dependencies
- 60 vulnerabilities
- **10 high severity**

There are **~700 tags** listed in Docker Hub official **ruby** repo...

...which one is "best"?

BASE IMAGE BEST PRACTICES

Generic is probably not what you want!

```
docker pull [ruby | python | ubuntu...]
```

- What framework version are you coding against? Do these containers match? Will they match tomorrow?
- Generally: good for playing around, but don't use for "real" work

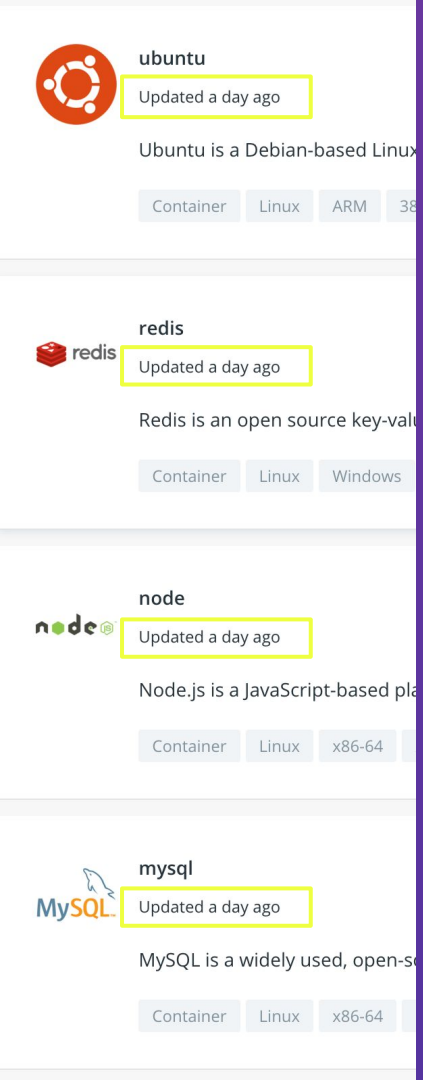
But `-slim` isn't automatically the right choice

- Vulns go away! 🥳
- BUT you "get to" manage all the build dependencies 😒

```
RUN apt-get update &&\
    apt-get install -y build-essential \
        patch ruby-dev zlib1g-dev liblzma-dev \
        libpq-dev libsqlite3-dev
```

MULTI-STAGE TO THE RESCUE!!!

```
FROM python:3.8-buster as builder
.
.
.
FROM python:3.8-buster-slim
COPY –from=builder <app stuff> .
```



GENERAL RECOMMENDATIONS

General “make life easier” things:

- A little bit of Sysadmin knowledge is unavoidable - pick an upstream distribution and use it for everything
- Pin to versioned images (at least Major, probably minor)

Getting rid of vulnerabilities

- Learn & love multi-stage builds!
- Let Docker / Red Hat / VMware (Bitnami)… do the heavy lifting!
- Rebuild often (clear your cache or use `--no-cache`)
- Move your pins every once in a while 📌

44%

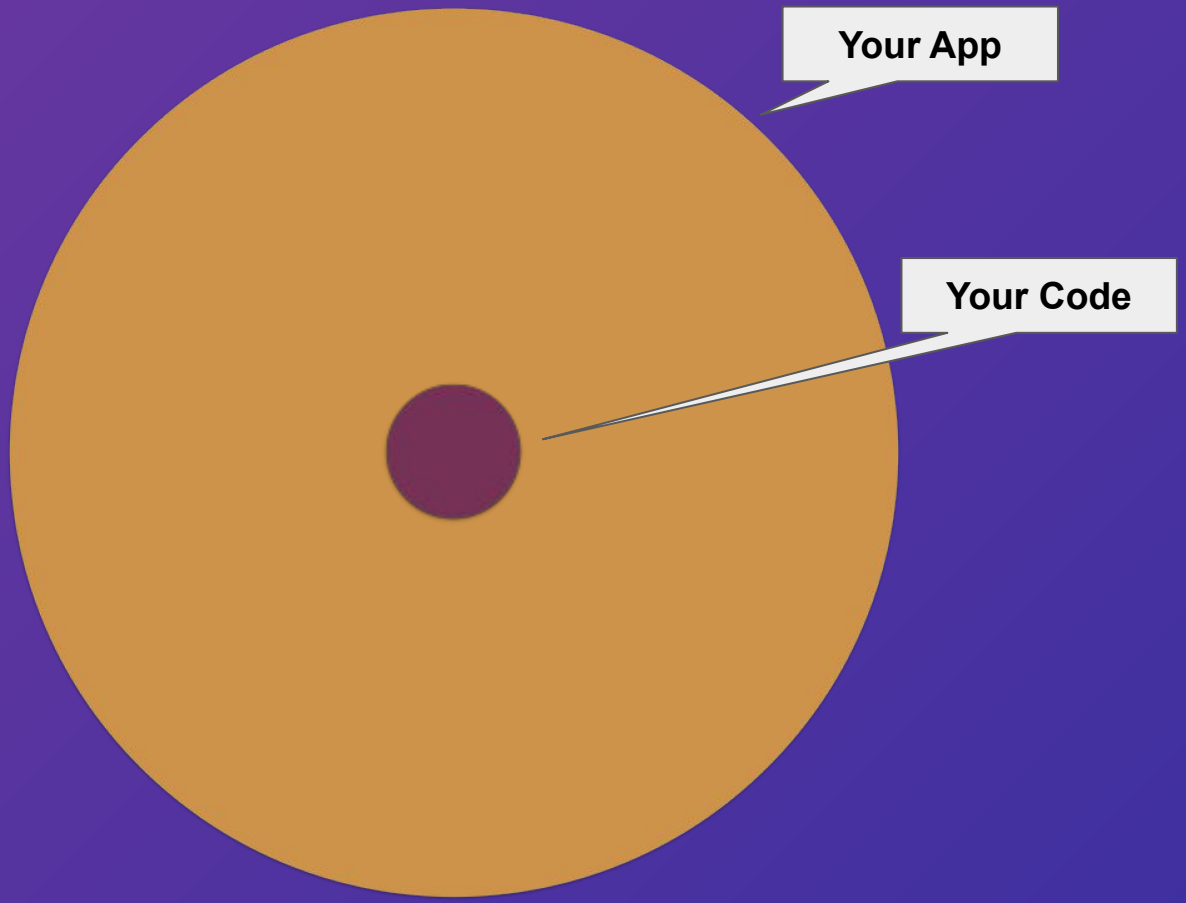
of docker image vulnerabilities can be fixed with newer base images

20%

of docker image vulnerabilities can be fixed just by rebuilding them


Your code
Your code dependencies
Your added layers
Base Image


Your code
Your code dependencies
Your added layers
Base Image

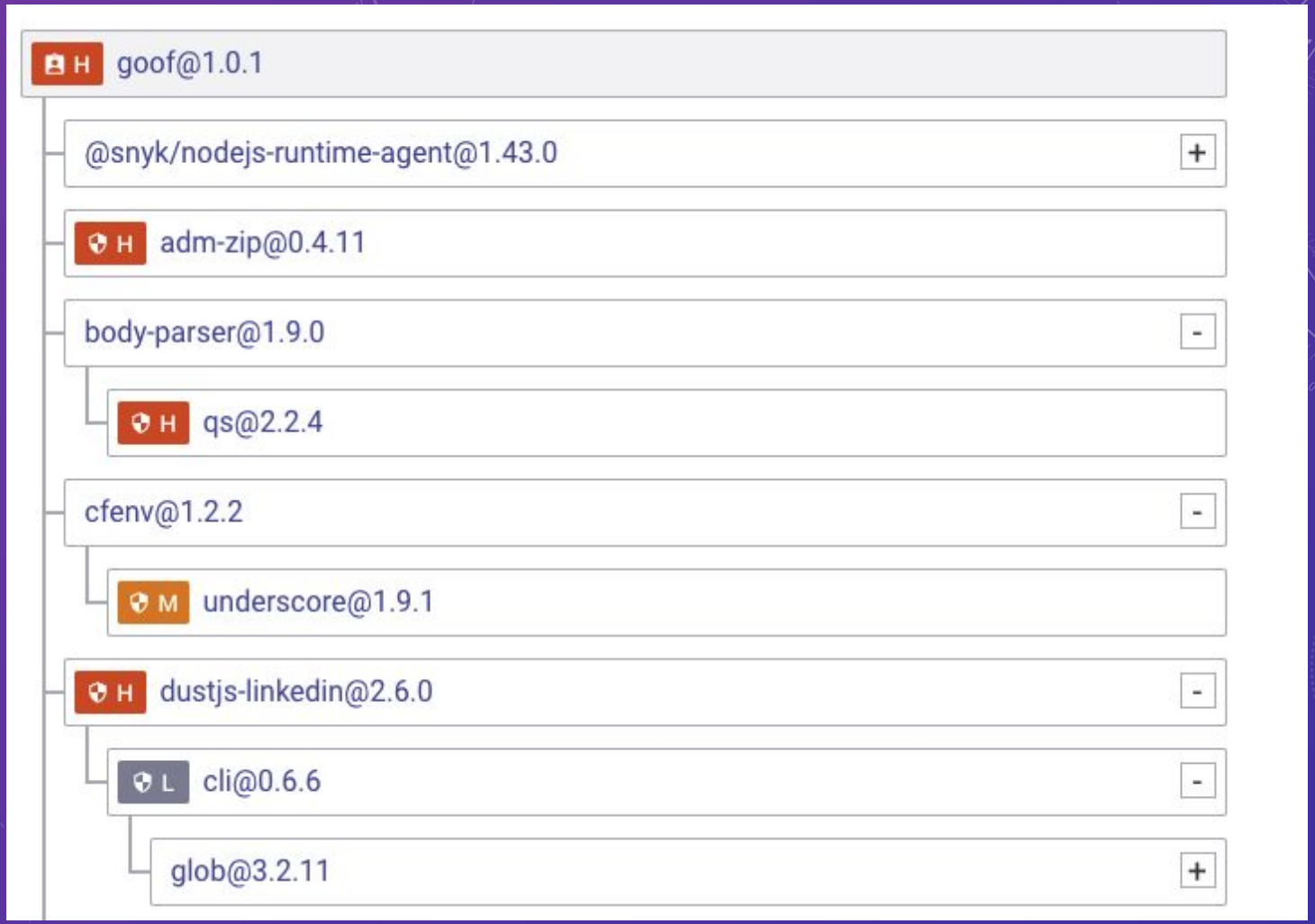

Your App
Your Code

```
},
 "dependencies": {
   "adm-zip": "0.4.7",
    "cookie-parser": "1.3.3",
   "dustjs-helpers": "1.5.0",
   "dustjs-linkedin": "2.5.0",
   "ejs": "1.0.0",
   "ejs-locals": "1.0.2",
   "errorhandler": "1.2.0",
   "express": "4.12.4",
   "express-fileupload": "0.0.5",
   "file-type": "^8.1.0",
   "humanize-ms": "1.0.1",
   "jquery": "^2.2.4",
   "lodash": "4.17.4",
   "marked": "0.3.5",
   "method-override": "latest",
   "moment": "2.15.1",
   "mongoose": "5.12.13",
       "st": "0.2.4",
   "stream-buffers": "^3.0.1",
   "tap": "^11.1.3"
 },
```

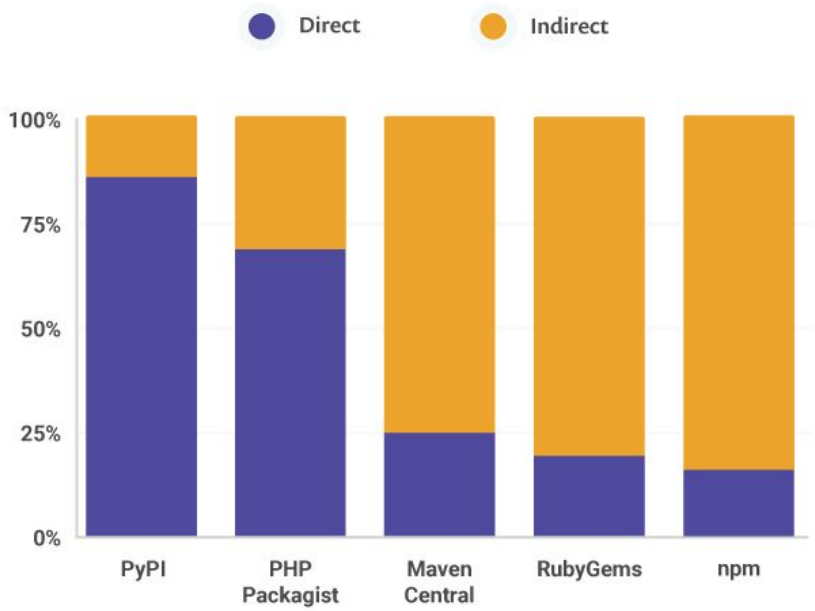
package.json

```
docopt == 0.6.1
keyring >= 4.1.1
coverage != 3.5
Mopidy-Dirble ~= 1.1
```

requirements.txt


H goof@1.0.1
@snyk/nodejs-runtime-agent@1.43.0 +
H adm-zip@0.4.11
body-parser@1.9.0 -
H qs@2.2.4
cfenv@1.2.2 -
M underscore@1.9.1
H dustjs-linkedin@2.6.0 -
L cli@0.6.6 -
glob@3.2.11 +

The direct and indirect dependency split across ecosystems

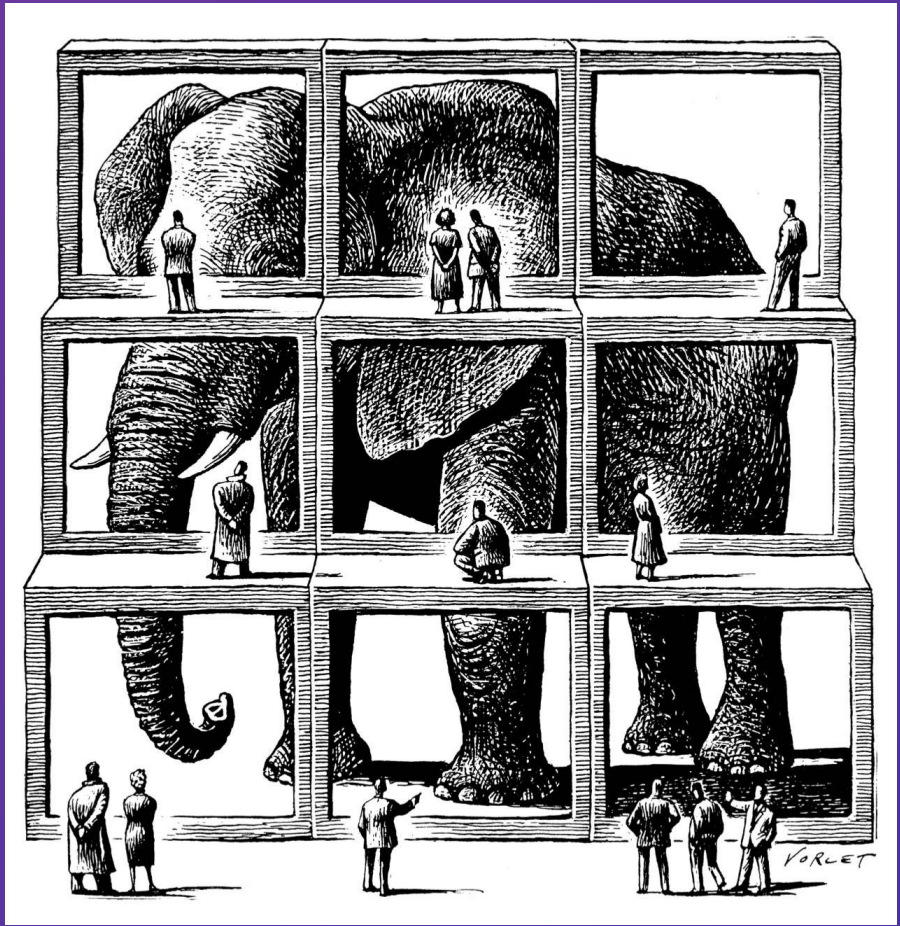


source: https://snyk.io/opensourcesecurity-2019


2
purpledobie/goof
93 H 84 M 448 L
View Report
demo
57 H 53 M 443 L
Tested 23 minutes ago
demo:/usr/src/goof/package.json
36 H 31 M 5 L
Tested 24 minutes ago


5
mattj-io/goof
270 H 239 M 487 L
View Report
Code analysis
3 H 5 M 0 L
Tested 11 hours ago
Dockerfile
230 H 196 M 469 L
Tested 13 hours ago
manifests/
goof-deployment.yaml
0 H 6 M 12 L
Tested 6 days ago
manifests/
goof-service.yaml
0 H 1 M 0 L
Tested 6 days ago
package.json
37 H 31 M 6 L
Tested 12 hours ago

Zero vulnerabilities in a container is almost impossible



Context matters ..

Base Score

9.8
(Critical)

Attack Vector (AV)

Network (N) | Adjacent (A) | Local (L) | Physical (P)

Attack Complexity (AC)

Low (L) | High (H)

Privileges Required (PR)

None (N) | Low (L) | High (H)

User Interaction (UI)

None (N) | Required (R)

Scope (S)

Unchanged (U) | Changed (C)

Confidentiality (C)

None (N) | Low (L) | High (H)

Integrity (I)

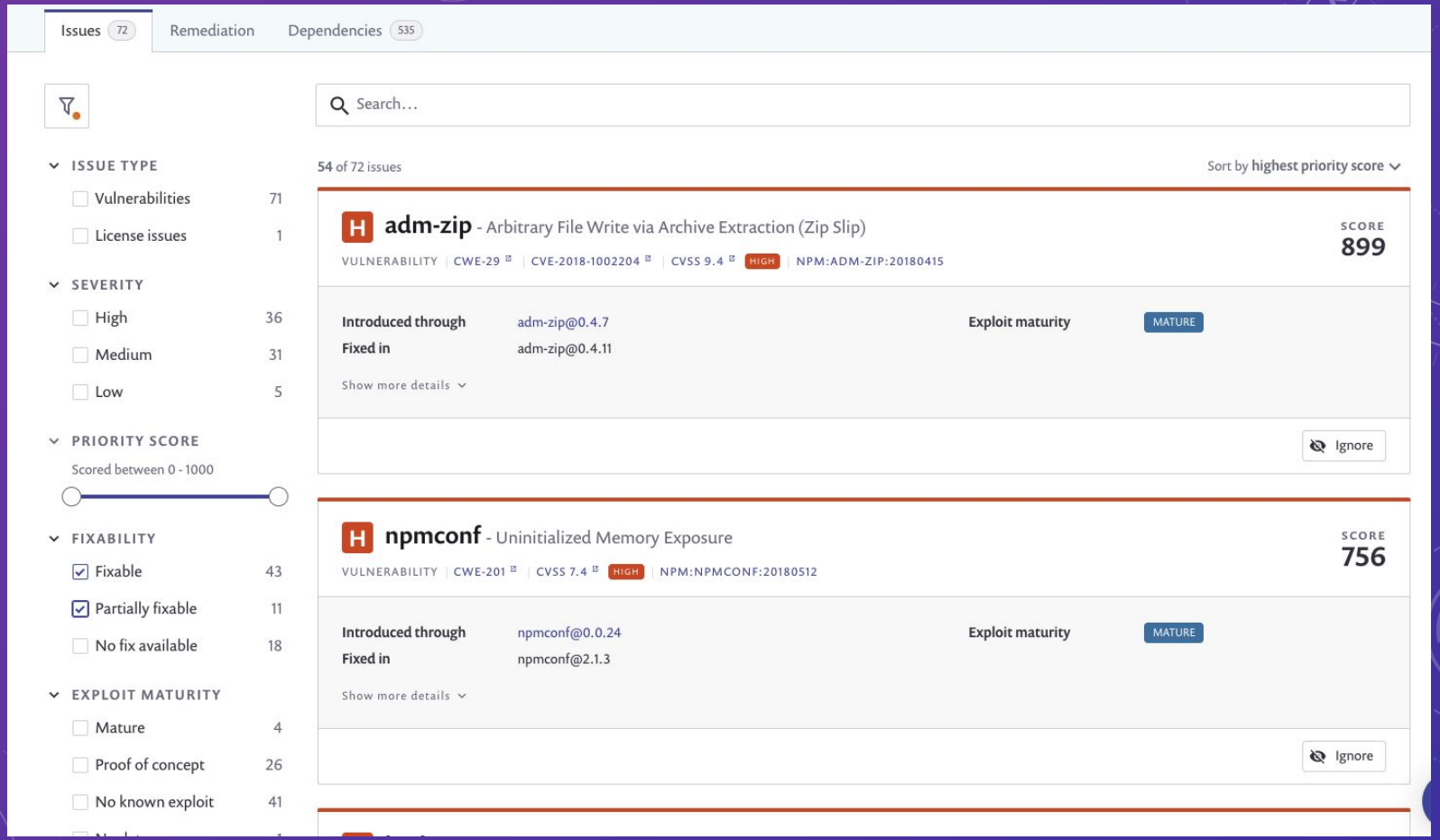
None (N) | Low (L) | High (H)

Availability (A)

None (N) | Low (L) | High (H)

Vector String - **CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H**

Understand your tools


Issues 72
Remediation
Dependencies 535
Search…
ISSUE TYPE
Vulnerabilities 71
License issues 1
SEVERITY
High 36
Medium 31
Low 5
PRIORITY SCORE
Scored between 0 - 1000
FIXABILITY
Fixable 43
Partially fixable 11
No fix available 18
EXPLOIT MATURITY
Mature 4
Proof of concept 26
No known exploit 41
54 of 72 issues
Sort by highest priority score
H adm-zip - Arbitrary File Write via Archive Extraction (Zip Slip)
VULNERABILITY | CWE-29 | CVE-2018-1002204 | CVSS 9.4 HIGH | NPM:ADM-ZIP:20180415
SCORE 899
Introduced through adm-zip@0.4.7
Fixed in adm-zip@0.4.11
Exploit maturity MATURE
Show more details
Ignore
H npmconf - Uninitialized Memory Exposure
VULNERABILITY | CWE-201 | CVSS 7.4 HIGH | NPM:NPMCONF:20180512
SCORE 756
Introduced through npmconf@0.0.24
Fixed in npmconf@2.1.3
Exploit maturity MATURE
Show more details
Ignore

- Mitigations
- Effort vs risk
- Filtering on CVSS

- No high CVE’s in production
- Nothing with a mature exploit
- Apply patches if there are any

Using base images

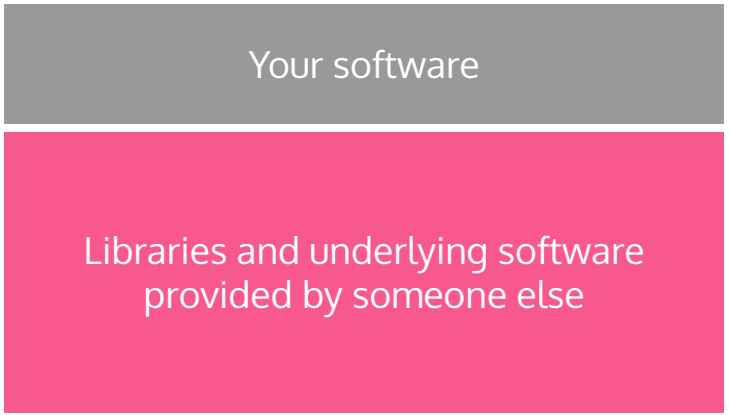
```
FROM ubuntu:latest
```

Standing on the shoulders of software giants

It's common with container images to start building on top of an existing base image that already has software you want.

This might be an operating system like `ubuntu`, `alpine` or `debian` or it could be a language like `python`, `ruby`, `node` or really anything else.

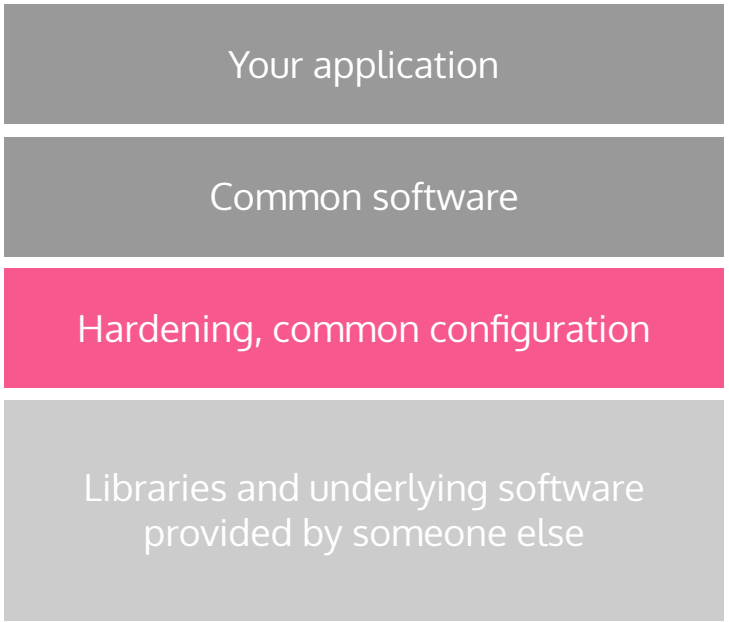
OK, technically often a parent image but hey.



Distinct responsibilities

Hardening, common configuration

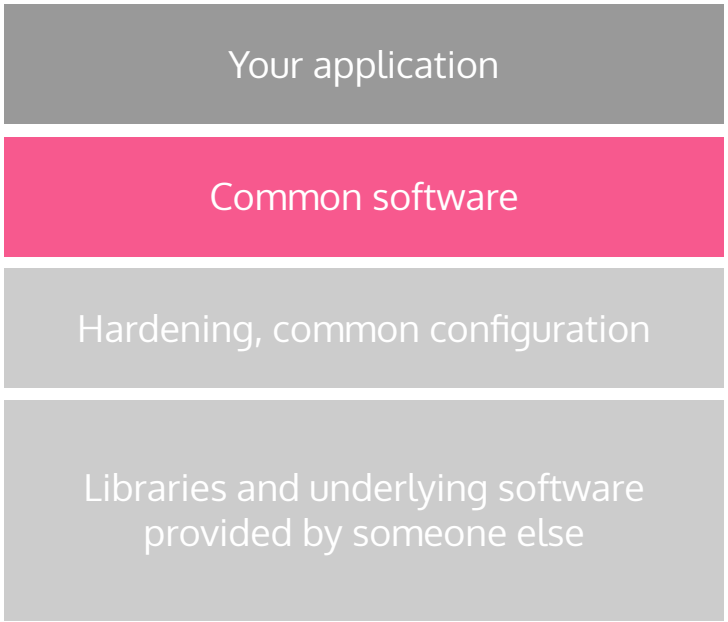
Your organization might have some common hardening or configuration changes or maybe metadata it wants to apply to all images in use by other teams. This is often intended to be common for all images used. Maybe you have `myorg/base`



Distinct responsibilities

Common software

Some organizations provide a layer of common software or middleware. This might be language or framework specific, say a separate image for Java (`myorg/java`) and another for Python (`myorg/python`).



Distinct responsibilities

Your application

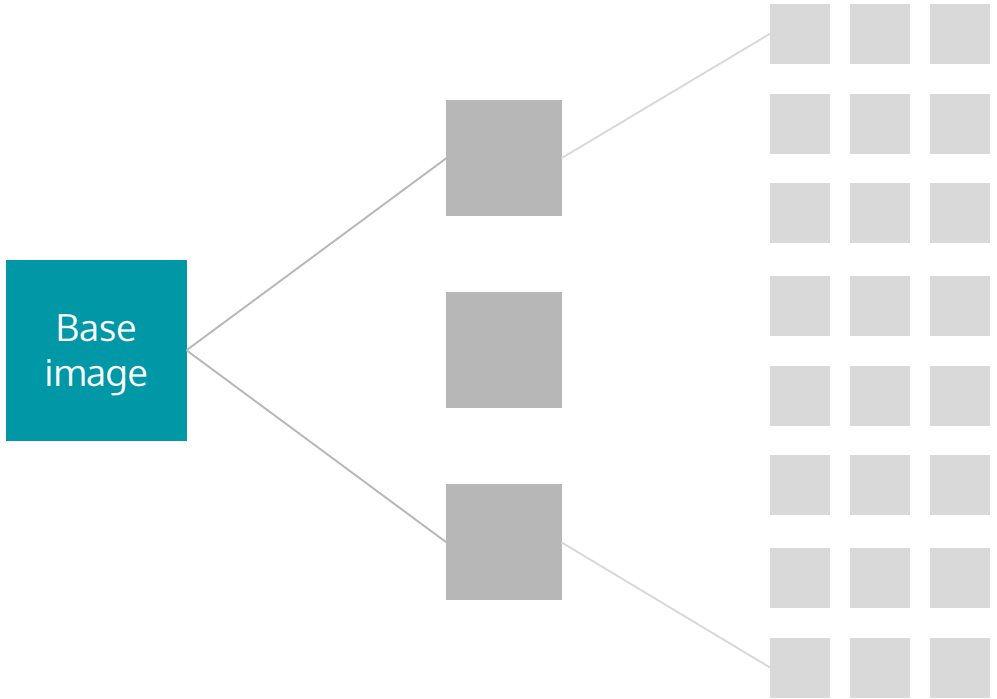
Finally the specifics of your application, whether in source or binary form. And metadata specific to the application.

Can you fix vulnerabilities once?

Scale vulnerability management

When considering container vulnerabilities, you want to be able to reason about vulnerabilities in images you're running, but also understand the overlap and source of those vulnerabilities.

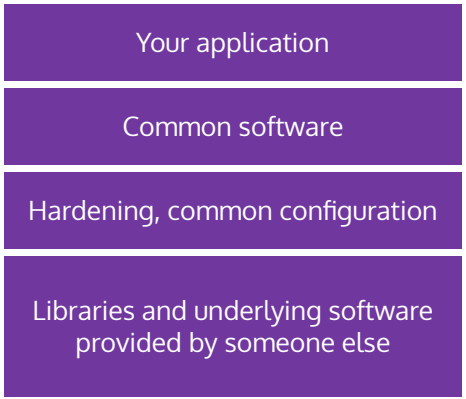
Can you address a vulnerability onc
and have it resolved everywhere?



Teams and responsibility

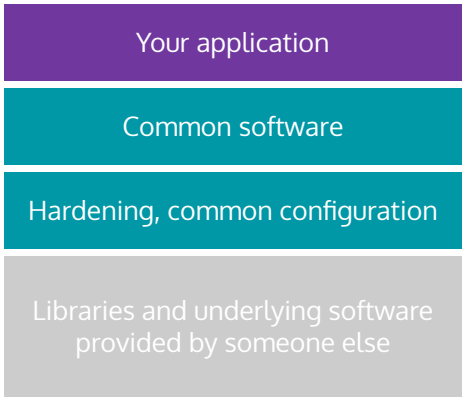
A worked example

Organizing into teams



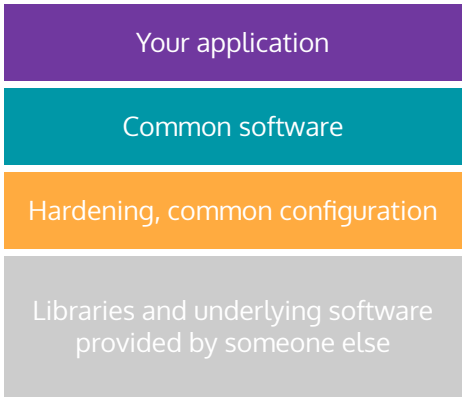
One team to rule them all

This could be the case when teams are completely independent, or when you have one central image team.



A base image team

A team which provides a standard set of approved base images for application teams to consume.



Separate base/security teams

Larger organizations might have teams with more distinct responsibilities, potentially with even more layers.

Pros and cons

One team to rule them all

PROS
Simple to understand responsibilities.

CONS
Potential for chaos if every team can do their own thing. One central team for ALL images likely to become a bottleneck.

A base image team

PROS
Able to build strong domain expertise in the center, ideally fix/triage issues once.

CONS
Needs some level of governance in order to ensure applications teams benefit from central expertise.

Separate base/security teams

PROS
Same as having a base image team, with the added advantage of deeper specialisms.

CONS
Coordination between additional teams can slow down the process.

Start with your own base image

```
$ cat base/Dockerfile
FROM python:3.6.0-slim
RUN apt-get update && apt-get install -y <all our critical hardening pkgs>

$ docker build -t myorg/base -f base/Dockerfile
```

Establish a baseline

What can downstream consumers ignore?

```
$ <container tool> --test myorg/base
...
  Introduced by your base image (python:3.6.0-slim)
  Fixed in: 5.28.1-6+deb10u1

✗ High severity vulnerability found in gnutls28/libgnutls30
  Description: Out-of-bounds Write
  Info: https://snyk.io/vuln/SNYK-DEBIAN10-GNUTLS28-609778
  Introduced through: gnutls28/libgnutls30@3.6.7-4+deb10u4, apt@1.8.2.1
  From: gnutls28/libgnutls30@3.6.7-4+deb10u4
  From: apt@1.8.2.1 > gnutls28/libgnutls30@3.6.7-4+deb10u4
  Introduced by your base image (python:3.6.0-slim)


Tested 111 dependencies for known issues, found 178 issues.
```

Watch out for new vulnerabilities

Generally, “fresh” images on Hub, RHT, etc have no FIXABLE vulns

- Your baseline establishes the things you can ignore
- But things get fixed…
- …and things get broken…
- **Rebuild & set a new baseline often!**

A middleware image

```
$ cat middleware/Dockerfile
FROM myorg/base

RUN apt-get update && apt-get install -y \
    gunicorn \
    sqlite3 \
    && rm -rf /var/lib/apt/lists/

$ docker build -t myorg/middleware -f middleware/Dockerfile
```

The sum of all vulnerabilities

```
$ <container tool> test myorg/middleware
...

Tested 127 dependencies for known issues, found 180 issues.
```

Vulnerabilities from the base image and the new instructions

But a different team is responsible for some of these, so let's reason about those separately.

Where did the issue come from?

There are really only 2 new vulns in middleware

```
$ <container tool> test myorg/middleware
```

Middleware team

Found 180 unique vulnerabilities for myorg/middleware

Package	Severity	ID	Issue	Installed	Fixed in
ncurses/libncurses5	HIGH	CVE-2017-10684	Out-of-Bounds	5.9+20140913-1+b1	5.9+20140913-1+deb8u1
ncurses/libncurses5	HIGH	CVE-2017-10685	Improper Input Validation	5.9+20140913-1+b1	5.9+20140913-1+deb8u1
sqlite3/libsqlite3-0	HIGH	CVE-2020-9794	Out-of-bounds Read	3.8.7.1-1+deb8u6	
sqlite3/libsqlite3-0	HIGH	CVE-2019-8457	Out-of-bounds Read	3.8.7.1-1+deb8u6	
ncurses/libncurses5	MEDIUM	CVE-2017-16879	Out-of-Bounds	5.9+20140913-1+b1	5.9+20140913-1+deb8u3
ncurses/libncurses5	MEDIUM	CVE-2017-13729	Out-of-Bounds	5.9+20140913-1+b1	5.9+20140913-1+deb8u1

...

Base image team

Base image vulnerabilities from myorg/base

Package	Severity	ID	Issue	Installed	Fixed in
apt/libapt-pkg4.12	HIGH	CVE-2019-3462	Arbitrary Code Injection	1.0.9.8.4	1.0.9.8.5
bzip2/libbz2-1.0	HIGH	CVE-2019-12900	Out-of-bounds Write	1.0.6-7+b3	1.0.6-7+deb8u1
glibc/libc-bin	HIGH	CVE-2018-1000001	Out-of-Bounds	2.19-18+deb8u7	
glibc/libc-bin	HIGH	CVE-2014-9761	Out-of-Bounds	2.19-18+deb8u7	

Your application images

```
$ cat app/Dockerfile
FROM myorge/middleware

EXPOSE 8080

WORKDIR /app
ADD app.py .

CMD ["gunicorn", "-w", "4", "app:app"]

$ docker build -t myorg/app -f app/Dockerfile
```

Visualising ownership

```
$ <container tool> test myorg/app

+-----------------------------------------------------
| Found 180 unique vulnerabilities for myorg/app
+-------------+-------------+-------------+-------------+-------------+-------------+
| Package     | Severity    | ID          | Issue       | Installed   | Fixed in    |
+-------------+-------------+-------------+-------------+-------------+-------------+

+-----------------------------------------------------------------------------------------------------------------------------------------------------+
| Base image vulnerabilities from myorg/middleware                                                                                                    |
+---------------------------------+-------------+-----------------+-----------------------------+---------------------+---------------------------+
| Package                         | Severity    | ID              | Issue                       | Installed           | Fixed in                  |
+---------------------------------+-------------+-----------------+-----------------------------+---------------------+---------------------------+
| apt/libapt-pkg4.12              | HIGH        | CVE-2019-3462   | Arbitrary Code Injection    | 1.0.9.8.4           | 1.0.9.8.5                 |
| bzip2/libbz2-1.0                | HIGH        | CVE-2019-12900  | Out-of-bounds Write         | 1.0.6-7+b3          | 1.0.6-7+deb8u1            |
| glibc/libc-bin                  | HIGH        | CVE-2018-1000001| Out-of-Bounds               | 2.19-18+deb8u7      |                           |

...
```

App team 🎉 🍻

Middleware team


ONE CYGNUS
ONE
UASC

```
apiVersion: v1
kind: Pod
metadata:
  name: nonroot-priv
spec:
  containers:
  - name: nonroot-priv
    image: mattjarvis/snyky
    volumeMounts:
      - mountPath: /chroot
        name: host
    securityContext:
      runAsUser: 999
      privileged: true
  volumes:
  - name: host
    hostPath:
      path: /
      type: Directory
```

Conclusions

If all you remember is...

Define your trust boundaries

Decide on a strategy

Start with the low hanging fruit

Thanks for listening

Sign up for free at snyk.io/signup

Twitter - @mattj_io
LinkedIn - https://www.linkedin.com/in/mattjarvis08

snyk