5 Security Best Practices for Production Ready Containers

Martin Wimpress

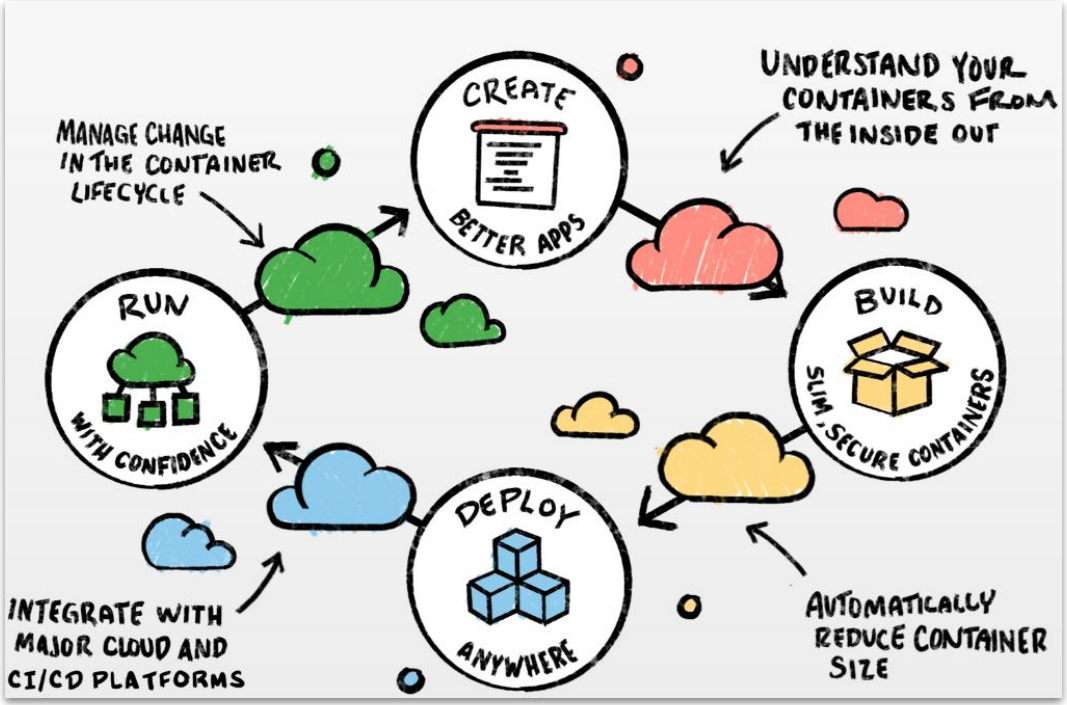
slim.ai

Slim.AI

Build secure containers, **faster**

Slim.AI was created to give developers the power to build safer cloud-native applications with less friction.

Martin Wimpress
Snr. Director of Developer Relations & Community



Agenda

Example app

Container best practices

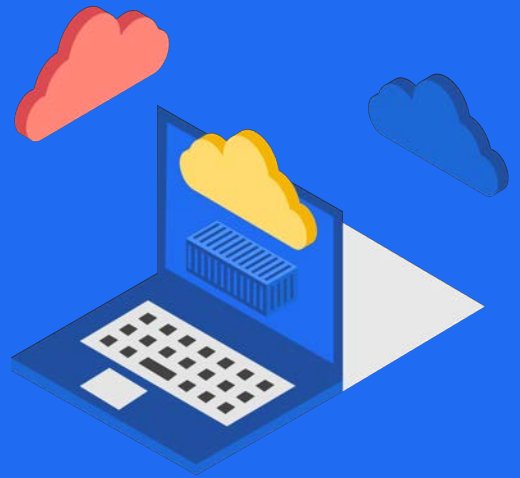
Container scanning & analysis

Choosing a base image

Container slimming

Example App

With multiple base images

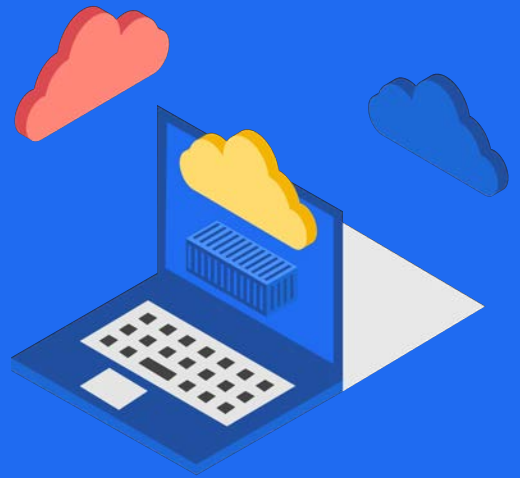


app.py

```python
#!/usr/bin/env python3

from flask import Flask, jsonify

app = Flask(__name__)
app.config['DEBUG'] = True

@app.route('/')
def index():
    return jsonify({'msg': 'Success'})

@app.route('/hello')
def hello():
     return jsonify({'msg': 'Hello World!'})

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=8008)
```

Container best practice

A better `Dockerfile`



A Better Dockerfile

Reputable base image
Official Python base image.

Layer construction
Minimize cache invalidation and optimise build performance.

ENTRYPOINT
Proper signal handling.

```
FROM python:3.9.13-slim-bullseye

RUN apt-get -y update && apt-get -y upgrade && \
    apt-get -y clean && rm -rf /var/lib/apt/lists/*

WORKDIR /app
COPY --chown=nobody:nobody app/requirements.txt ./
RUN pip3 install --no-cache-dir -r requirements.txt
COPY --chown=nobody:nobody app/app.py ./

USER nobody
EXPOSE 8008
ENTRYPOINT ["python3","app.py"]
```

USER nobody

root by default
If you do not specific a `USER`, your app runs as `root` by default.

nobody
An unprivileged system account available in most base images.

Limit impact
Significantly minimizes impact if your container is compromised.

```
USER nobody
```

Pinned “distro” & language versions

`python:latest`
No source of truth.
Unpredictable.

`python:3.9.13`
No language fixes or improvements.
Reproduced with ease and reliability.

```
FROM python:3.9.13-slim-bullseye
```

Apply updates

Base image latency
Base images can be days behind updates already available via package repositories.

Use a stable base
Using a stable base and “pinned” language versions will prevent unexpected upgrades.

```
RUN apt-get -y update && apt-get -y upgrade && \
    apt-get -y clean && rm -rf /var/lib/apt/lists/*
```

Layer caching

`RUN` text is cached
The text of the `RUN` command is used to determine if the cache should be used.

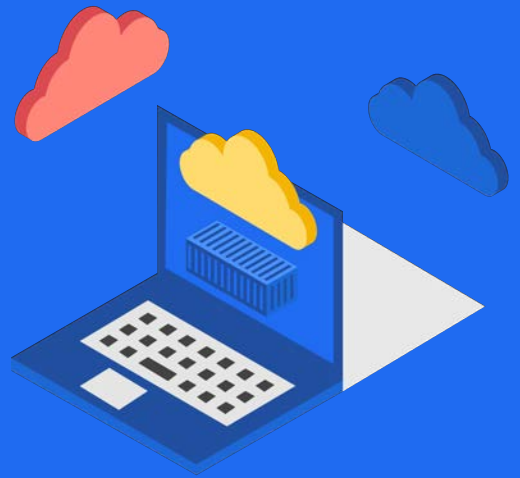
Caches may be insecure
Caches may contain old insecure packages even after updates have been published.

Build processes
Your normal build process and a daily fresh rebuild using `--pull --no-cache`

```
docker build --pull --no-cache -f app:latest .
```

Container scanning & analysis

Know what is in your container



Container scanning

Vulnerability scanning
Scan your containers for known vulnerabilities.

Generate SBOMs
Know what is inside your containers.

Add to build pipelines
Review routinely so you know what you're really shipping to production.

```
docker scan -f Dockerfile app:latest

docker sbom app:latest
```

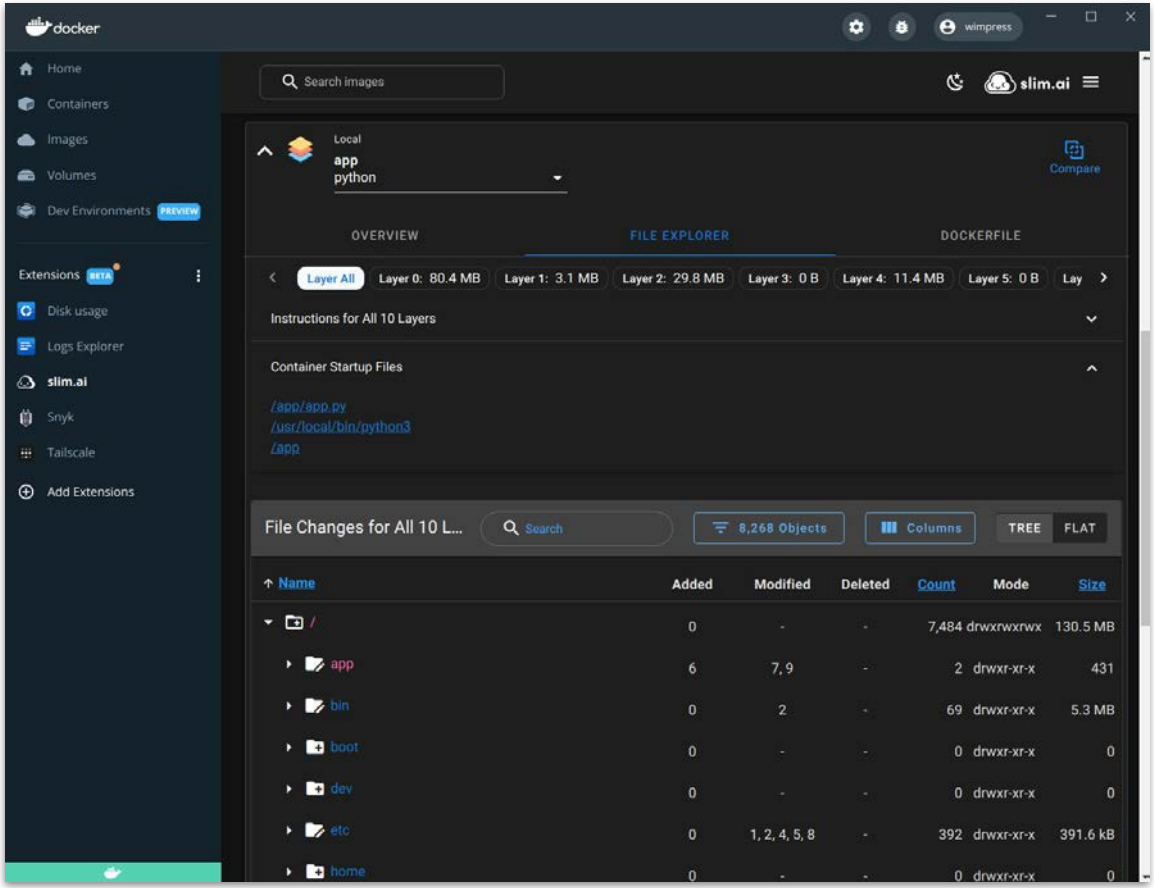
Container analysis

Understand

Knowing what's in a container is critical to securing your software supply chain.

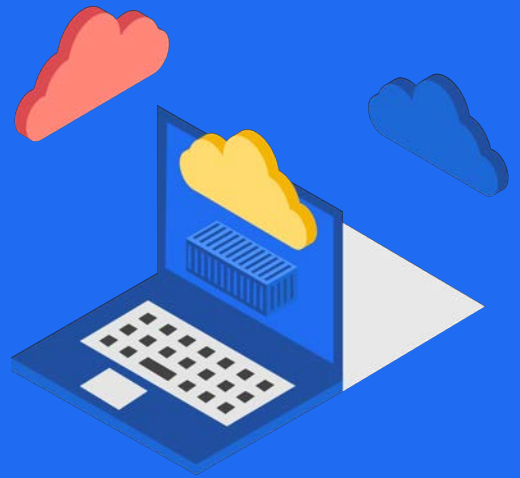
Inspect

Slim.AI lifts the veil on container internals so you can analyze, optimize, and compare changes before deploying your cloud-native apps.



Choosing a base image

Start with a good security posture



A “slim” base image

How much smaller?

The regular “fat” `python:3.9.13-bullseye` image is 915MB.

Smaller images

Slim container images are faster to deploy (lower size) and faster to start (fewer files).

	Size	
`python:3.9.13-alpine3.15`	51MB	🥇
`gcr.io/distroless/python3`	54MB	🥈 (Multistage)
`ubuntu:22.04`	78MB	🥉 (No Python)
`python:3.9.13-slim-bullseye`	125MB	

Fewer packages

Recommended packages

Most system package install default to installing all recommended packages.

Fewer packages

Fewer packages generally results in smaller images with a reduced attack surface.

```
RUN apt-get -y --no-install-recommends install
python3-minimal python3-pip


RUN dnf --nodocs -y install
--setopt=install_weak_deps=False python3
```

App by image size

Best practise works

Already significantly smaller than the regular 915MB `python:3.9.13-bullseye` base image!

“Smaller is safer”

So the saying goes.

	Size	
`python:3.9.13-alpine3.15`	62MB	🥇
`gcr.io/distroless/python3`	72MB	🥈
`Python:3.9.13-slim-bullseye`	135MB	🥉
`ubuntu:22.04`	139MB	

App by vulnerability count

Commercial vendor SLAs

Commercially backed Linux vendors commit to security SLAs. Community projects are best efforts.

Alpine has issues

Python, Node and some other languages, can result in significantly slower builds and introduce runtime bugs. Great for Go and Rust however.

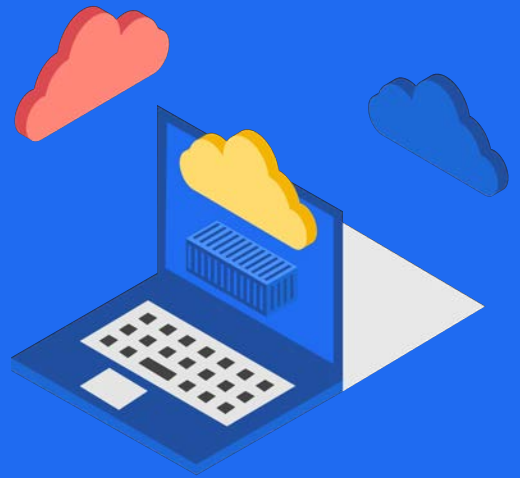
	Total	Crit	High	
python:3.9.13-alpine3.15	0	0	0	🥇
ubuntu:22.04	15	0	0	🥈
gcr.io/distroless/python3	46	3	7	🥉
python:3.9.13-slim-bullseye	84	13	3	

What if…

Low complexity of Ubuntu and the security profile and size of Alpine? 🤔💭

Container Slimming

Optimise and minify your containers



DockerSlim & Slim.AI

Optimise
DockerSlim and Slim SaaS can automatically optimize your container images.

Free & Open Source
DockerSlim is free and open source software available from GitHub.

https://dockersl.im

https://slim.ai

```
docker-slim build --tag app:latest
```

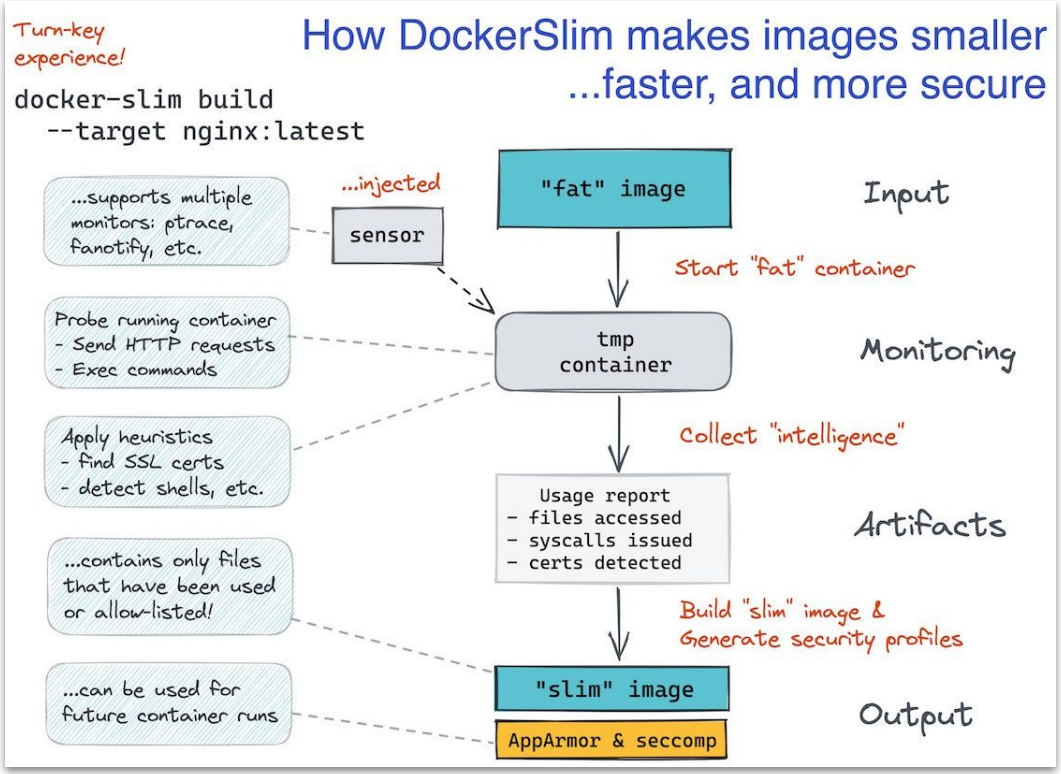
How does DockerSlim work?

Analysis

`docker-slim` optimizes containers by understanding what your application actually needs using various analysis techniques.

New single layer image

Creates a new single layer image using only the required files from the original *"fat"* image.



Why?

Ship “No Code”

Only ship into production what your app requires. Slim containers can be up to 30X smaller.

Faster

Slim container images are faster to deploy (lower size) and faster to start (fewer files).

Cost savings

Slim container images can be less expensive to store and transfer.

Security

Slim containers reduce the attack surface and vulnerability count. Unnecessary shells, tools, utilities and libraries are entirely removed.

App by slimmed image size

Slim all the things!

In most cases there are significant size reductions to be had slimming any container image, regardless of build technique and base image used.

	Fat	Slim	Reduced
python:3.9.13-alpine3.15	62MB	20MB	3.13X
gcr.io/distroless/python3	72MB	22MB	3.05X
python:3.9.13-slim-bullseye	135MB	23MB	5.95X
ubuntu:22.04	139MB	25MB	5.06X

Slimmed App by vulnerability count

Component searches

Analysing slimmed containers for vulnerable is currently a manual task, as the meta data scanning tools use is no longer available.

Only takes a few minutes (at most) using Slim.AI SaaS or Slim.AI Docker Extension.

	Total
python:3.9.13-alpine3.15	0
ubuntu:22.04	0

Conclusions

What did we learn?

slim.ai

Conclusions

- Follow container best practice
- `USER` should be unprivileged
- Pin your base image
- Use a stable base image & apply updates
- Be mindful of layer caching
- Container scanning and analysis are essential
- Do not install recommended packages
- Linux vendors have security SLAs
- Assess your base image options
- You can slim Alpine and Distroless containers
- Slimming significantly reduces vulnerabilities & attack surface

slim.ai

@SlimDevOps

Keep building.

DISCORD