Building Containers for Python Applications

Moshe Zadka – https://cobordism.com

2021

Acknowledgement of Country

Belmont (in San Francisco Bay Area Peninsula)
Ancestral homeland of the Ramaytush Ohlone people

What is good

What is good

- To crush your enemies

What is good

- To crush your enemies
- To see them driven before you

What is good

- To crush your enemies
- To see them driven before you
- Um, wrong slides

What is good

- Fast

What is good

- Fast
- Small

What is good

- Fast
- Small
- Secure

What is good

- Fast
- Small
- Secure
- Usable

Specifying the requirements

Let's be more concrete

- Keep up to date

Specifying the requirements

Let's be more concrete

- Keep up to date
- Reproducible builds

Specifying the requirements

Let's be more concrete

- Keep up to date
- Reproducible builds
- No compilers in prod

Specifying the requirements

Let's be more concrete

- Keep up to date
- Reproducible builds
- No compilers in prod
- Keep size (reasonably) small

Up to date

Install security updates

Up to date

Install security updates
But when?

Up to date

Install security updates
But when? Depends!

Reproducible builds

Same code gives same results

Reproducible builds

Same code gives same results
...mostly

No compilers in prod

A common anti-pattern

No compilers in prod

A common anti-pattern

...surprisingly easy to get wrong!

Size

- Diminishing returns

Size

- Diminishing returns
- Cost savings

Support binary wheels

Installing and building

Support binary wheels

Installing and building
Faster

Support binary wheels

Installing and building
Faster
Simplifies images

Not run as root

General hygiene

Minimal privileges

Especially avoid permissions to pip install

Fast rebuilds

Responsiveness!

Base OS

The distro wars are back?

Base - size

Most modern distros have a decent minimal server

Base - size

Most modern distros have a decent minimal server
...but Debian is easiest to get smallest.

Base - LTS/support

Usually around 5 years

Base - LTS/support

Usually around 5 years
Gives you time to upgrade!

Base - Volatility

How much change?
Security? Backports? Fixes?

Debian

LTS: 5 years
Conservative

Ubuntu

LTS: 5 years
(Universe, Multiverse, etc...)
Fairly conservative

Alpine (probably not)

Uses musl, not manylinux compatible

Rolling releases (probably not)

Up to date, but...

Rolling releases (probably not)

Up to date, but...
updates can change major versions!

CentOS

Rolling release!

How to get Python?

So many options...

Not system Python

Distros aim Python at distro packages

Not system Python

Distros aim Python at distro packages
not user programs.

Appropriate repositories

Famous examples: deadsnakes PPA for Ubuntu

pyenv

Builds and installs Python

python-build

Builds and installs Python

Source

```
RUN configure [...]
RUN make
RUN make install
```

Source

```
RUN configure [...]
RUN make
RUN make install
```

Build from source + Debian?

Source

```
RUN configure [...]
RUN make
RUN make install
```

Build from source + Debian?python: images are basically that!

Trade-offs

Control vs. Work vs. Problems

Versions

Support multiple for upgrade path

Versions

Support multiple for upgrade path
2-3

Container multistage build (quick recap)

Only one stage output

Container multistage build (quick recap)

Only one stage output
other stages help

FROM

Use previous stage as starting image

COPY –from

Copy files from previous stage

Stages a as modules

```
FROM ubuntu as security-updates
RUN add-apt-repository ppa:deadsnakes/ppa
RUN apt-get update
RUN apt-get upgrade

FROM security-updates as with-38
RUN apt-get install python3.8

FROM security-updates as with-39
RUN apt-get install python3.9
```

Separate build and runtime

Especially when building from source!

Separate build and runtime

Especially when building from source!

```
FROM ubuntu as builder
# install build dependencies
# build Python into /opt/myorg/python

FROM ubuntu as as runtime
COPY --from=builder \
       /opt/myorg/python \
       /opt/myorg/python
```

Optimizing layers

Put everything under /opt/myorg
Use one COPY --from=...

Optimizing size

After building Python, remove:

- Tests
- Builder dependencies (in runtime)
- ....and more

Binary wheels

- Build with builder
- Copy to runtime
- Install in virtual environment

Binary wheels (alt)

- Build with builder
- Install in virtual environment
- Copy virtual environment to runtime

Patchelf

Used to make wheels self-contained
Newest version needed

Auditwheel

Use pip to install

Self-contained binary wheels

Run

```
auditwheel repair --platform linux_x86_64
```

Self-contained binary wheels

Run

```
auditwheel repair --platform linux_x86_64
```

No need for binary dependencies!

Portable binary wheels

- Oldest supported?

Portable binary wheels

- Oldest supported?

Example:

```
auditwheel repair --platform manylinux_2_27_x86_64
```

Generating binary wheels

Build instructions in docs

Generating binary wheels

Build instructions in docs
Build dependencies

Optimizing layers

Reduce copies

Optimizing layers

Reduce copies
Prep

Optimizing caching

Where to build wheel?

Optimizing caching

Where to build wheel?
What invalidates caching?

Conclusion

- Wrong easier than right

Conclusion

- Wrong easier than right
- But right is amazing

Conclusion

- Wrong easier than right
- But right is amazing
- Think before you docker

Further Resources

Itamar's series – https://pythonspeed.com/docker/