Mastering Alpine Linux

31.05.2017

Maxime Vidori

Alpine? Never heard of it...

What’s in the box?

- ☐ **provide a package manager and a small footprint** (v3.5 ~4MB).
- ☐ Based on busybox and musl-libc.
- ☐ Can be used as a distribution and come with a grsec kernel.

How can this help me?

- ☐ Easier to understand and deploy.
- ☐ Force you to invest time in your system, and production environment.
- ☐ Reduce security risks by mastering your toolchain, **no more third party unknown containers!**

Alpine? musl libc

musl

lightweight, *fast*, *simple*, *free*, and strives to be *correct* in the sense of standards-conformance and safety.

- ☐ Replacement for the **glibc**, works most of the time.
- ☐ **~600KB** vs **~8MB** for complete .so set.
- ☐ Some softwares will not compile (I am looking at you **systemd**).
- ☐ You can still install it, but this is sketchy and not recommended outside a chroot (see the documentation).

Alpine? busybox

busybox

The Swiss Army knife of Embedded Linux

- Simple binary with minimal versions of common UNIX utilities (rm, ls, ...).
- Minimal size (**~2MB**)
- Primarily designed as a recovery tool.
- Used by major projects such as Debian for the installation.

Alpine? apk

apk is the tool used to install, upgrade, or delete software on a running alpine system.

- introduce some other dependencies: libcrypto, libssl, libz.
- more convenient than a basic scratch image.
- good tooling and documentation.

When not to use it

- The use of *musl-libc* as the core library can cause some dependencies to not build.
- When building big images the small footprint is no longer an advantage (cross compiler can be really huge).
- Package library is not exhaustive (10GB big, for main and community), this is not a debian distribution, if a lot of dependencies are involved do not use it.

Tooling tips and tricks!

- virtual package, remove dev dependencies easily
  (golang containers can have size reduced from **~200MB** to **~11MB**)
- create local mirror for packages (only ~5GB for the main repo), allow rapid offline builds, push your custom packages.
- custom packages Alpine package description file is based on Gentoo Linux **ebuilds**, an easy way to package is to check Archlinux AUR for examples.

Tooling tips and tricks!

- ☐ Start your mirror server

```
docker run -p "8080:80" demo/server
```

- ☐ Build your image

```
docker build -t demo/base:1.0 .
```

```
FROM alpine:3.5
RUN echo "http://bridge.ip:8080" > /etc/apk/repositories
ADD repo-key.rsa.pub /etc/apk/keys
```

- ☐ Use your pipeline!

```
FROM demo/base:1.0
RUN apk add --no-cache -t dependencies hello ...
```

Just use **minikube** or **compose**!

Demo!

… What could possibly go wrong?

Conclusion

- When building containers, think size, think network, think build time.
- Be sure to use the right tools for the right thing.
- Before starting new tools, look at what already exists.

That' s all folks

Questions?

IxDay/mastering-alpine