A COMPLETE guide on how to make Docker images even smaller  
  
2022-03-17T04:52:59.244Z Highlights by John Donne with Liner

**\_\_\_\_\_**  
  
A COMPLETE guide on how to make Docker images even smaller

**\_\_\_\_\_**  
  
Table of contents  
   
   
 Use a smaller base image or libc  
 Make the image context smaller  
 Fully minimize and tidy up image layers  
 Splitting up processes and services for breaking up images  
 Takeaways

**\_\_\_\_\_**  
  
Choose a small base image.  
The most used base image, ubuntu, is already very optimized. However, the ubuntu image changed over time and it is worth upgrading. Here is a list of the latest images:  
  
ubuntu:14.04, 197MB  
ubuntu:16.04, 135MB  
ubuntu:18.04, 63.2MB  
ubuntu:20.04, 72.8MB  
ubuntu:21.04, 80MB  
ubuntu:22.04, 79MB

**\_\_\_\_\_**  
  
alpine:latest, 5.59MB  
busybox:glibc, 4.79MB  
busybox:musl, 1.43MB  
busybox:uclibc, 1.43MB

**\_\_\_\_\_**  
  
For the very adventurous, one can also usescratch as a base image. Which has not only 0 MB (or in words ZERO MEGABYTES) but is also a no-op (a no-operation), i.e. it does not even add a layer to your image. If you do not want to go that far, there are base images specific to the application runtime you are using, e.g. for Go and Java, without adding any operating system specific files. One of these is called distroless by Google.

**\_\_\_\_\_**  
  
By default, Docker adds all files as the build context when doing a docker build. Hence, if you have a monorepo like us, your build context is HUGE.

**\_\_\_\_\_**  
  
A .dockerignore file exactly defines which files are used as the build context. The default .dockerignore file we are using at Symflower starts with the following lines.  
\*  
.\*  
These two lines instruct Docker to ignore every visible and hidden file and directory by default. After that we are adding exceptions that should be included. This allows us to exactly define which files should be part of the build context.

**\_\_\_\_\_**  
  
we can, for example, use the following new image definition:  
FROM ubuntu  
  
RUN apt-get update \  
 && apt-get install -y --no-install-recommends curl \  
 && apt-get autoremove -y \  
 && apt-get purge -y --auto-remove \  
 && rm -rf /var/lib/apt/lists/\*  
This already yields a reduction by 36MB, which results in an image size of 87MB.

**\_\_\_\_\_**  
  
APT::Install-Suggests "0"  
APT::Install-Recommends "0";  
We went with the /etc/apt/apt.conf solution, and this immediately made a difference. That is, we obviously had multiple apt-get install commands without the --no-install-recommends option. The monster CI image went down from 6.09GB to 5.9GB, a saving of about 190MB.

**\_\_\_\_\_**  
  
Looking at the biggest directory /usr, we saw that 408MB are used by the Go installation, 101MB by the NodeJS installation, 527MB by LLVM, 418MB by Java and 202MB by Chromium. In total 1656MB that cannot be reduced any further. This problem continues with other parts of our container image: we see files that we can remove, e.g. documentation, but in the end most files are necessary to build or run certain jobs of our CI. We hit the point, where going forward means painstakingly questioning every package and file we have in our container image.

· Highlighted Source : https://share.getliner.com/5tCegCwYuB/  
· Original Source : https://symflower.com/en/company/blog/2022/complete-guide-on-shrinking-container-images/