
SciLifeLab

UNDERSTANDING CONTAINER IMAGES

DATA CENTRE - DEVOPS CHAPTER

ALVARO REVUELTA

SciLifeLab

PRESENTATION STRUCTURE

1. Introduction
2. Image Structure
3. How layers are generated
4. Optimize images
5. Conclusion

SciLifeLab

1. INTRODUCTION

- The presentation will be intercalated with a demo using the tool Dive and docker command line.

Motivation

- Bridging Knowledge: **Complex technical stack**.
- Understanding the importance of formation in virtualization and container technology for **everyone**.

SciLifeLab

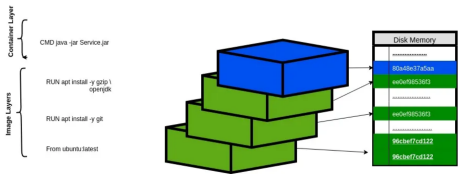
1. INTRODUCTION

- Images are **immutable**. Once an image is created, it can't be modified.
- Container images are composed of **layers**. Each layer represents a set of file system changes that add, remove, or modify files.

SciLifeLab

2. IMAGE STRUCTURE

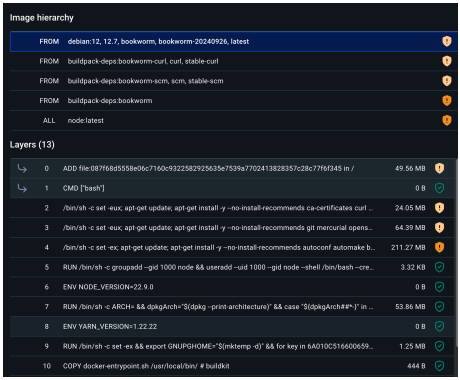
- Images consist of **multiple layers** stacked on top of each other.
- Layers are **file system changes** (e.g., file additions, deletions, updates).
- Layers are **read-only**; only the top container layer is **writable** at runtime.
- Layers are merged to form a **Unified File System** (Full explanation would need another presentation).



(**Source**: UnionFS : A File System of a Container)

3. HOW IMAGES ARE GENERATED

- Layers are **reusable** in subsequent builds, because they are cached.
- **FROM**: Setting the base image (starting point)
- **RUN**: Run a command (e.g., installing software)
- **COPY/ADD**: Adding files
- **CMD/ENTRYPOINT**: Setting the default command for running the container
- RUN, ADD and COPY generate new layers.



(**Source**: DockerHub)

4. OPTIMIZE IMAGES

Minimizing the Number of Layers:

- Combine multiple RUN commands into one (e.g., chaining commands using &&).
- Removing unnecessary files during the build process (e.g., apt-get clean).
- Avoid using COPY to add files that won't be needed during runtime.

Using Smaller Base Images:

- Alpine Linux vs. Ubuntu: Trade-offs between size and functionality.
- Multi-stage builds: Using one image for building the application and another for the final image (stripped of development tools).

Caching Layers Efficiently:

- Take advantage of layer caching by ordering Dockerfile instructions logically.
- Place less frequently changed commands earlier in the Dockerfile to maximize caching.

Removing Unnecessary Packages and Files:

- Remove temporary files, build dependencies, or logs to reduce image size.
- Clean up the file system after installations (e.g., rm -rf /var/lib/apt/lists/*).

SciLifeLab

4. OPTIMIZE IMAGES - DISTROLESS

- If every change in the file system creates a new layer. Can there be an empty image?
- Yes! Introducing **FROM SCRATCH**
- Images built using this base are called **distroless**.
- By reducing the image size, you are also reducing the **attack surface** for possible vulnerabilities.

SciLifeLab

4. OPTIMIZE IMAGES - DISTROLESS

- Even thought there is a considerable increase in security and performance, images built this way introduce a considerable **development overhead**.
- For learning purposes they are pretty useful, but for production ready environment maybe not the best.
- Google realized this some years back and developed their own **suite of distroless images** (link to Google Distroless repository).

```
# use ubuntu as base image
FROM ubuntu as build-env

# install build-essential package to compile the source code
RUN apt update && apt install -y build-essential

# copy the source code
COPY hello.c hello.c

# Compile and generate binary
RUN gcc -o helloWorld hello.c

# use distroless container to run the program
FROM SCRATCH

# copy the required libraries to run the program
COPY --from=build-env  /lib/aarch64-linux-gnu/libc.so.6 /lib/aarch64-linux-gnu/libc.so.6
COPY --from=build-env  /lib/ld-linux-aarch64.so.1 /lib/ld-linux-aarch64.so.1

# copy the /etc/passwd
COPY --from=build-env  /etc/passwd /etc/passwd

# tmp directory
COPY --from=build-env  /tmp /tmp

# copy binary executable to new layer
COPY --from=build-env ./helloWorld ./helloWorld
```

(**Source**: own)

SciLifeLab

4. OPTIMIZE IMAGES - DISTROLESS

- In the quest for optimizing images, some projects have appeared.
- Going back to the introduction and understanding the need of having dedicated time for learning for companies that want to use Cloud Native technologies.

- Ubuntu Chiselled images (link).
- COPA project (link).

SciLifeLab

5. CONCLUSSION

Key Takeaways

- Importance of understanding image layers and their role in optimization.
- Best practices for reducing image size and improving efficiency.
- Tools and techniques for analyzing and improving images.

Q&A ?

SciLifeLab