Distributing Docker container applications: Legal challenges and solutions

Compliant container applications with the OSADL Docker Base Image

Caren Kresse

Open Source Automation Development Lab (OSADL) eG

COMPACT OSADL ONLINE LECTURES

What are Docker containers?

- Virtualization on the operating system or application level
- "Docker" is the trademark of the company Docker®, Inc.
- A **Docker container** (*i.e.* a running instance) is created from a **Docker image** (*i.e.* a collection of software on disk).
- A Docker image can be created from a **Dockerfile** (*i.e.* instructions on how to assemble the image).
- The Docker engine manages containers.

Why are containers so popular?

- All **dependencies**, data and configurations are contained in an **isolated environment**.
- Distribution and deployment of software is simplified.
- **Compatibility** issues are avoided.

Why are containers so popular?

- All **dependencies**, data and configurations are contained in an **isolated environment**.
- Distribution and deployment of software is simplified.
- **Compatibility** issues are avoided.
- **BUT:** The possibility to "freeze" a system in a particular state, removes the need to stay updated and within the requirements of a distribution’s package management system, which can have impacts on security.

How is a container image made up?

- Stacked layers that contain different components, functionalities and modifications.

How is a container image made up?

- Stacked layers that contain different components, functionalities and modifications.
- For example:

Layer 1: Base layer → System requirements: C library, package manager, shell …

How is a container image made up?

- Stacked layers that contain different components, functionalities and modifications.
- For example:

Layer 2: Application	→ Applications and their libraries
Layer 1: Base layer	→ System requirements: C library, package manager, shell …

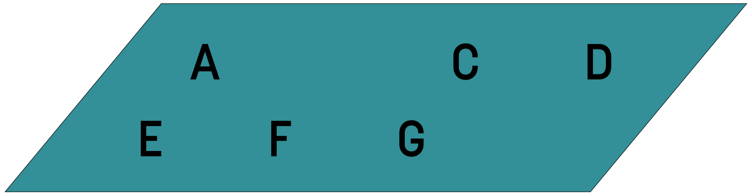
How is a container image made up?

- Stacked layers that contain different components, functionalities and modifications.
- For example:

Layer 3: Customization	→	Removal and modification of components
Layer 2: Application	→	Applications and their libraries
Layer 1: Base layer	→	System requirements: C library, package manager, shell …

How is a container image made up?

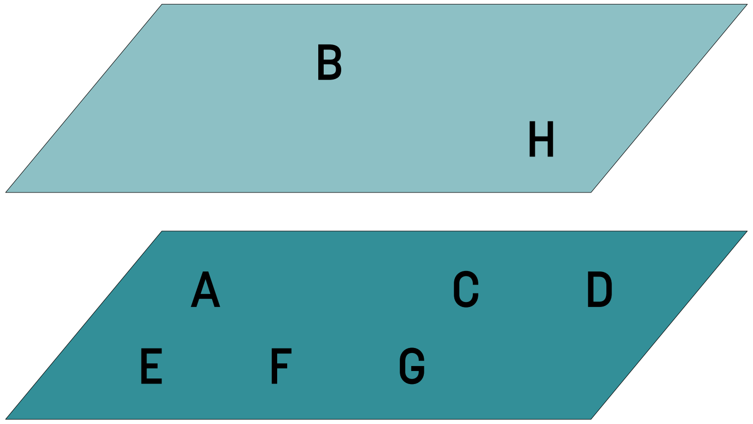
- For example:



→ Layer 1: Base image

How is a container image made up?

- For example:

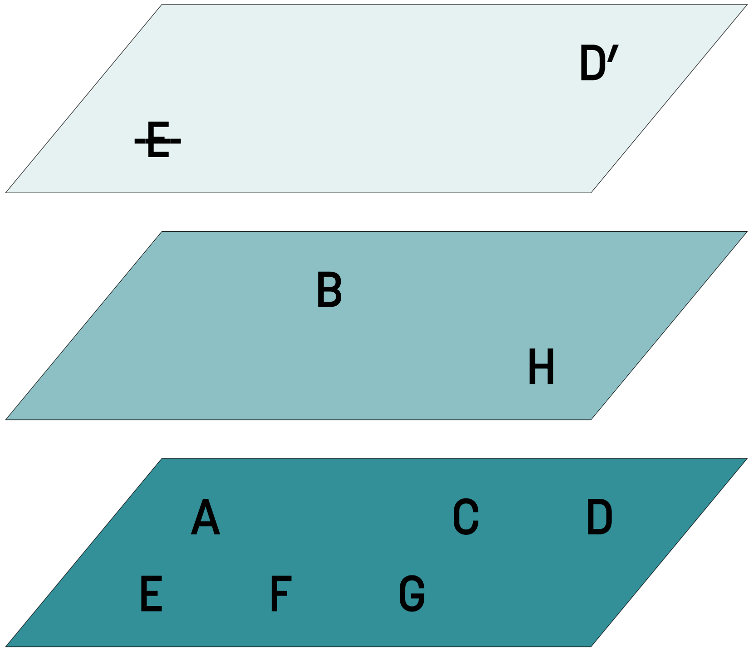


→ Layer 2: Add files B and H

→ Layer 1: Base image

How is a container image made up?

- For example:



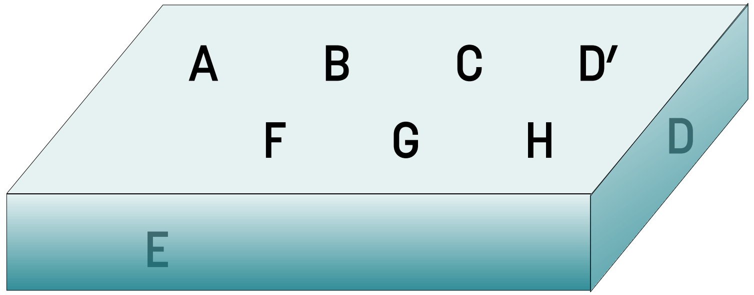
→ Layer 3: Remove file E and modify file D

→ Layer 2: Add files B and H

→ Layer 1: Base image

What does the resulting container look like?

- The run-time view shows only the result, but all components are still present inside the container:



What is inside the container image?

- Most containers include **Free and Open Source software** (FOSS)
- When **copying and distributing** FOSS, license obligations have to be fulfilled, *e.g.*
  - **Information obligations:** License texts, legal notices, warranty disclaimer
  - **Disclosure obligations:** Source code and build instructions
  - **Licensing obligations:** Adapting company documents, licensing own development correctly if a derivative work with software under a copyleft license is created

What is the challenge with containers ...

... from a **legal compliance** point of view?

- It is not obvious what software is distributed within the container.
  - But license obligations must be fulfilled for <u>all</u> software that is distributed.
- Public container repositories often do not provide compliance information.
  - But license obligations generally remain with the distributor not with the owner of the repository.

What about Dockerfiles?

- Dockerfiles are **scripts** from which a Docker engine can assemble a container image.
- Software is **downloaded** from public or private repositories.
- Dockerfiles can be **distributed** instead of container images.

Who is responsible for license compliance when distributing a Dockerfile?

- The person who **controls** the distribution.

→ **Distributing a Dockerfile = Distributing software**

Who is responsible for license compliance when distributing a Dockerfile?

- The person who **controls** the distribution.

  → **Distributing a Dockerfile = Distributing software**

- Difficulties:
  - Repositories mostly do not contain compliance material (*e.g.* license information, source code)
  - Commands in a Dockerfile may point to different software at different times (*e.g.* version "latest")

A possible exception?

- FOSS licenses of the container image do not impose obligations for programs that are required to run the container but are not distributed, *e.g.* the operating system or the Docker engine.
- Container base images consist of essential **system requirements** and **system libraries**.

→ The exception could be extended to **base images**.

A possible exception!

→ No license obligations apply for a base image distributed via Dockerfile

A possible exception!

→ No license obligations apply for a base image distributed via Dockerfile

Conditions:

- No modifications.
- The base image is offered in a license compliant way from the specified repository.

A possible exception!

→ No license obligations apply for a base image distributed via Dockerfile

Conditions:

- No modifications.
- The base image is offered in a **license compliant way** from the specified repository.

The OSADL Docker Base Image

- **Minimal image based on a Linux distribution:**
  - Essential GNU tools, bash, glibc, package manager, networking functionalities
- **License compliance material:**
  - Source code of all packages
  - License texts
  - Legal notices

How to create a compliant base image (1)

Initial installation: Create a chroot environment and install a base image into it (using *debootstrap* on Debian, *apk.static* on Alpine)

Set up (from within the chroot): Add source repository, set nameserver

Source download: Use the package manager to list installed packages and their version and to download the corresponding source packages

How to create a compliant base image (2)

License scan: Exit chroot and export the source packages; unpack and patch them and run a license scan to create compliance material (license texts, copyright notices, acknowledgments)

Information obligation: Create and display legal information

Create container image: Create a container image from the chroot directory:

```
tar cpf - . | docker import - [Repo/Image:Version]
```

Immediate or delayed source code delivery

Immediate source code delivery

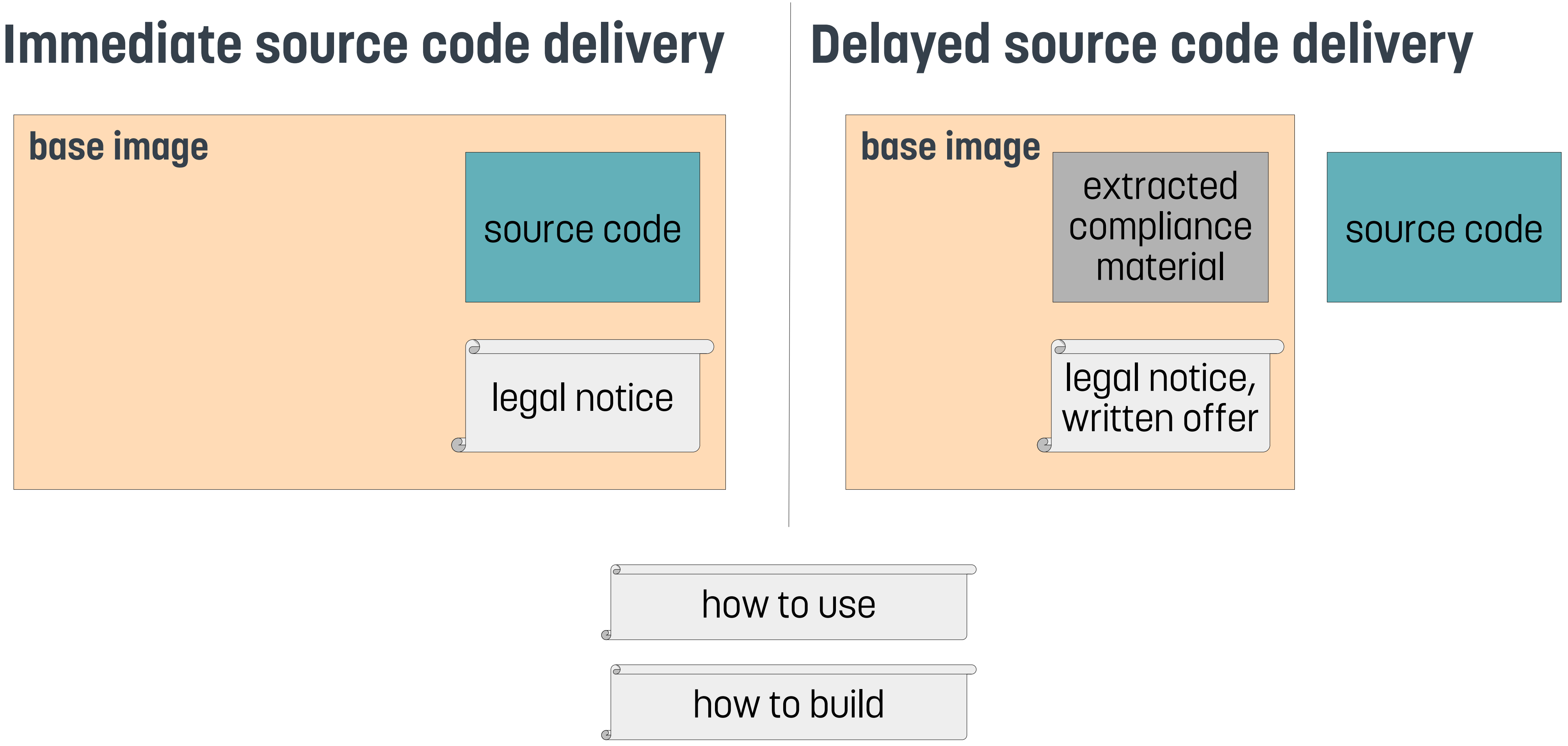
- All source code and a note with legal information are included in the image.
- Also available on Docker Hub: https://hub.docker.com/repository/docker/osadl/osadl-docker-base-image

Delayed source code delivery

- Extracted compliance material and a note with legal information and written offer are included in the image.
- The source packages including rebuild instructions for later delivery are provided separately.

Instructions on how the base image is created and how it can be used are given in each case.

Immediate or delayed source code delivery


Immediate source code delivery
base image
source code
legal notice
Delayed source code delivery
base image
extracted compliance material
source code
legal notice, written offer
how to use
how to build

Variants of the OSADL Docker Base Image

- Available at *osadl.org/base-image*
- Variants
  (in each case for immediate or delayed source code delivery):
  - Alpine 3.14 amd64
  - Debian 10 (buster) amd64
  - Debian 11 (bullseye) amd64
- Additional variants can be created and made available **on request**.

Building an application on top of the OSADL Docker Base Image

To exemplify the process of building a **customized container image** on top of the OSADL Docker Base Image

- Base image: Debian 10 (buster)
- Sample application: hello-world program of the *ImmediateC* project (*https://www.osadl.org/?id=382*, *https://github.com/JohnWulff/immediatec*)
- Build process: via Dockerfile

ImmediateC container image: Dockerfile

```
FROM osadl-docker-base-image/debian-buster_amd64:v1.2_binary-only
```

Base image

ImmediateC container image: Dockerfile

```
FROM osadl-docker-base-image/debian-buster_amd64:v1.2_binary-only

RUN apt-get install -y make wget sudo gcc libc6-dev bison flex perl \
  libperl5.28 perl-tk libperl4-corelibs-perl ssh xauth net-tools
```

Base image

Installing dependencies from distro

ImmediateC container image: Dockerfile

```
FROM osadl-docker-base-image/debian-buster_amd64:v1.2_binary-only

RUN apt-get install -y make wget sudo gcc libc6-dev bison flex perl \
  libperl5.28 perl-tk libperl4-corelibs-perl ssh xauth net-tools

RUN cd /usr/src && wget \
 http://mirror.23media.de/cpan/modules/by-authors/id/D/DE/DEWEG/Time-HiRes-
01.20.tar.gz && tar -xzf Time-HiRes-01.20.tar.gz && cd Time-HiRes-01.20 \
 && perl Makefile.PL && make && make test && make install
```

Base image

Installing dependencies from distro

Downloading, building and installing additional dependencies

ImmediateC container image: Dockerfile

```
FROM osadl-docker-base-image/debian-buster_amd64:v1.2_binary-only

RUN apt-get install -y make wget sudo gcc libc6-dev bison flex perl \
  libperl5.28 perl-tk libperl4-corelibs-perl ssh xauth net-tools

RUN cd /usr/src && wget \
 http://mirror.23media.de/cpan/modules/by-authors/id/D/DE/DEWEG/Time-HiRes-
01.20.tar.gz && tar -xzf Time-HiRes-01.20.tar.gz && cd Time-HiRes-01.20 \
 && perl Makefile.PL && make && make test && make install

RUN cd /usr/src && wget --no-check-certificate \
 https://github.com/JohnWulff/immediatec/archive/refs/tags/icc_3.6.tar.gz \
 && tar -xzf icc_3.6.tar.gz && cd immediatec-icc_3.6/src && ./configure \
 && make && make install
```

Base image

Installing dependencies from distro

Downloading, building and installing additional dependencies

Downloading, building and installing *ImmediateC*

ImmediateC container image: Dockerfile

```
FROM osadl-docker-base-image/debian-buster_amd64:v1.2_binary-only

RUN apt-get install -y make wget sudo gcc libc6-dev bison flex perl \
  libperl5.28 perl-tk libperl4-corelibs-perl ssh xauth net-tools

RUN cd /usr/src && wget \
 http://mirror.23media.de/cpan/modules/by-authors/id/D/DE/DEWEG/Time-HiRes-
01.20.tar.gz && tar -xzf Time-HiRes-01.20.tar.gz && cd Time-HiRes-01.20 \
 && perl Makefile.PL && make && make test && make install

RUN cd /usr/src && wget --no-check-certificate \
 https://github.com/JohnWulff/immediatec/archive/refs/tags/icc_3.6.tar.gz \
 && tar -xzf icc_3.6.tar.gz && cd immediatec-icc_3.6/src && ./configure \
 && make && make install

RUN echo -e "X11UseLocalhost no\nPermitRootLogin yes"
>>/etc/ssh/sshd_config \
 && echo root:root | chpasswd
```

Base image

Installing dependencies from distro

Downloading, building and installing additional dependencies

Downloading, building and installing *ImmediateC*

Additional customization

ImmediateC container image: Dockerfile

```
FROM osadl-docker-base-image/debian-buster_amd64:v1.2_binary-only

RUN apt-get install -y make wget sudo gcc libc6-dev bison flex perl \
  libperl5.28 perl-tk libperl4-corelibs-perl ssh xauth net-tools

RUN cd /usr/src && wget \
 http://mirror.23media.de/cpan/modules/by-authors/id/D/DE/DEWEG/Time-HiRes-
01.20.tar.gz && tar -xzf Time-HiRes-01.20.tar.gz && cd Time-HiRes-01.20 \
 && perl Makefile.PL && make && make test && make install

RUN cd /usr/src && wget --no-check-certificate \
 https://github.com/JohnWulff/immediatec/archive/refs/tags/icc_3.6.tar.gz \
 && tar -xzf icc_3.6.tar.gz && cd immediatec-icc_3.6/src && ./configure \
 && make && make install

RUN echo -e "X11UseLocalhost no\nPermitRootLogin yes"
>>/etc/ssh/sshd_config \
 && echo root:root | chpasswd

RUN rm -rf /var/lib/apt/lists/* \
 && rm -f /usr/src/icc_3.6.tar.gz /usr/src/Time-HiRes-01.20.tar.gz
```

Base image

Installing dependencies from distro

Downloading, building and installing additional dependencies

Downloading, building and installing *ImmediateC*

Additional customization

Cleaning up

ImmediateC container image: Dockerfile

```
FROM osadl-docker-base-image/debian-buster_amd64:v1.2_binary-only

RUN apt-get install -y make wget sudo gcc libc6-dev bison flex perl \
  libperl5.28 perl-tk libperl4-corelibs-perl ssh xauth net-tools

RUN cd /usr/src && wget \
 http://mirror.23media.de/cpan/modules/by-authors/id/D/DE/DEWEG/Time-HiRes-
01.20.tar.gz && tar -xzf Time-HiRes-01.20.tar.gz && cd Time-HiRes-01.20 \
 && perl Makefile.PL && make && make test && make install

RUN cd /usr/src && wget --no-check-certificate \
 https://github.com/JohnWulff/immediatec/archive/refs/tags/icc_3.6.tar.gz \
 && tar -xzf icc_3.6.tar.gz && cd immediatec-icc_3.6/src && ./configure \
 && make && make install

RUN echo -e "X11UseLocalhost no\nPermitRootLogin yes"
>>/etc/ssh/sshd_config \
 && echo root:root | chpasswd

RUN rm -rf /var/lib/apt/lists/* \
 && rm -f /usr/src/icc_3.6.tar.gz /usr/src/Time-HiRes-01.20.tar.gz

CMD /etc/init.d/ssh start && ifconfig && /bin/bash
```

Base image

Installing dependencies from distro

Downloading, building and installing additional dependencies

Downloading, building and installing *ImmediateC*

Additional customization

Cleaning up

Specifying entry point

ImmediateC container image: Dockerfile

```
FROM osadl-docker-base-image/debian-buster_amd64:v1.2_binary-only

RUN apt-get install -y make wget sudo gcc libc6-dev bison flex perl \
  libperl5.28 perl-tk libperl4-corelibs-perl ssh xauth net-tools

RUN cd /usr/src && wget \
 http://mirror.23media.de/cpan/modules/by-authors/id/D/DE/DEWEG/Time-HiRes-
01.20.tar.gz && tar -xzf Time-HiRes-01.20.tar.gz && cd Time-HiRes-01.20 \
 && perl Makefile.PL && make && make test && make install

RUN cd /usr/src && wget --no-check-certificate \
 https://github.com/JohnWulff/immediatec/archive/refs/tags/icc_3.6.tar.gz \
 && tar -xzf icc_3.6.tar.gz && cd immediatec-icc_3.6/src && ./configure \
 && make && make install

RUN echo -e "X11UseLocalhost no\nPermitRootLogin yes"
>>/etc/ssh/sshd_config \
 && echo root:root | chpasswd

RUN rm -rf /var/lib/apt/lists/* \
 && rm -f /usr/src/icc_3.6.tar.gz /usr/src/Time-HiRes-01.20.tar.gz

CMD /etc/init.d/ssh start && ifconfig && /bin/bash
```

Every line is a new layer in the final image!

Building the image and running the container

```
# cd ImmediateC-container

# docker build -t immediatec .

# docker image ls
REPOSITORY     TAG IMAGE   ID             CREATED           SIZE
immediatec     latest      64bcc21b1861   5 seconds ago     619MB


# docker run -it immediatec:latest
[…]
inet 172.17.0.2  netmask 255.255.0.0  broadcast 172.17.255.255
[…]
```

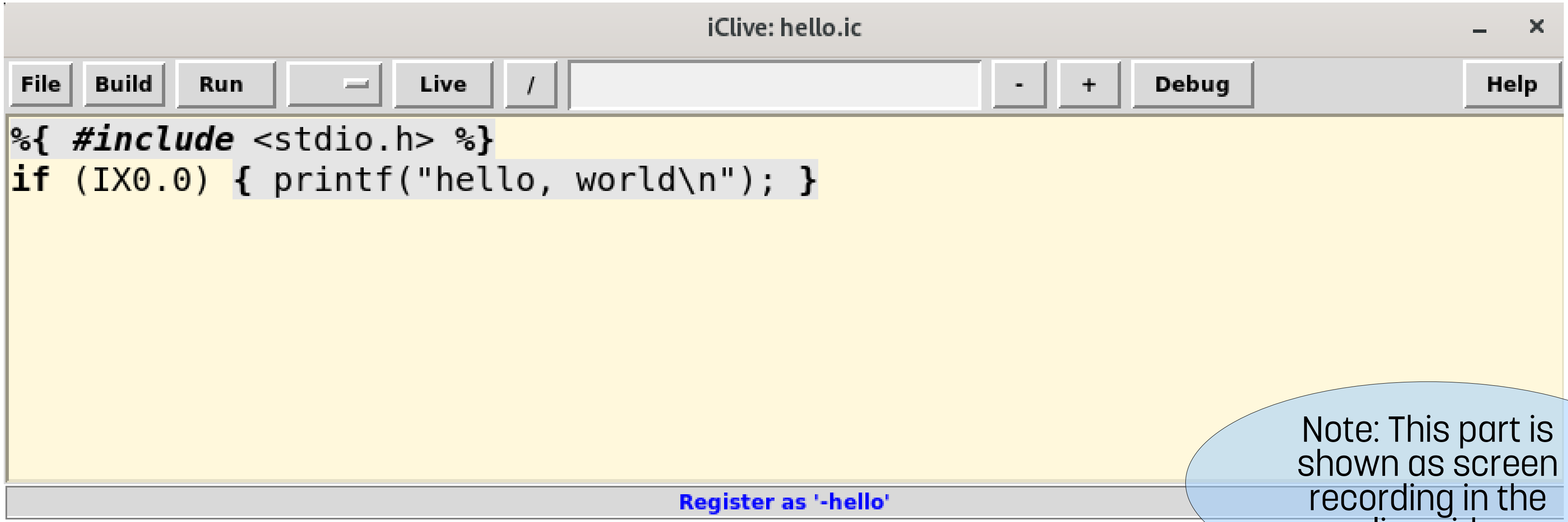
Note: This part is shown as screen recording in the online video.

COOL COMPACT OSADL ONLINE LECTURES

OSADL

Testing the application from remote machine

```
$ ssh -X root@172.17.0.2 'cd /usr/src/immediatec-icc_3.6/src; iClive hello.ic'
$ iCserver -z -k -A iCbox &
```



```
%{ #include <stdio.h> %}
if (IX0.0) { printf("hello, world\n"); }
```

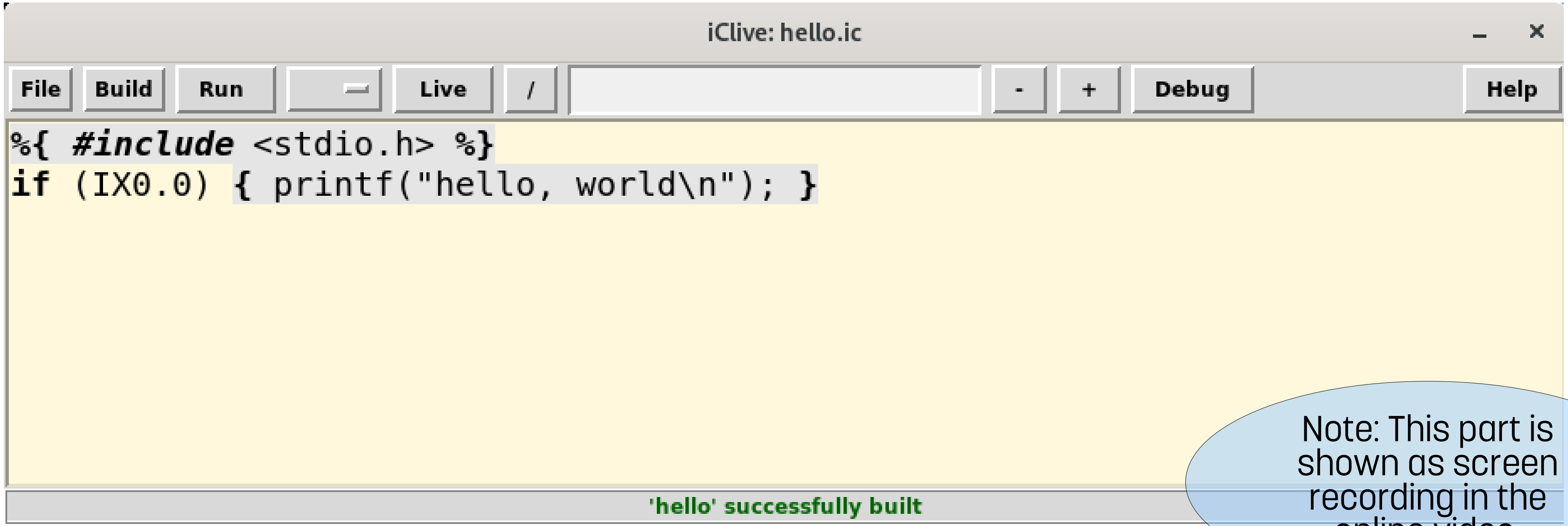
Register as '-hello'

Note: This part is shown as screen recording in the online video.

Testing the application: Building

Build → Build executable

```
$ iCmake -fsAd200 hello.ic 2>&1
```



```
%{ #include <stdio.h> %}
if (IX0.0) { printf("hello, world\n"); }
```

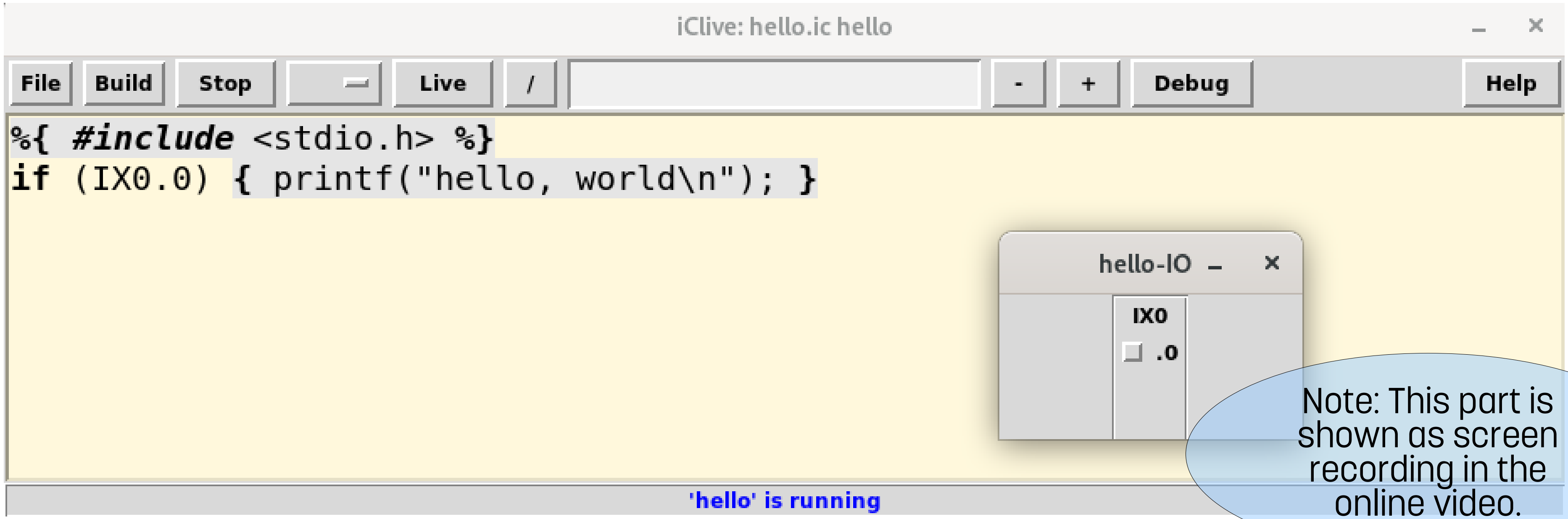
'hello' successfully built

Note: This part is shown as screen recording in the online video.

Testing the application: Running

Run

```
$ ./hello -d1000 &
$ iCbox -z -nhello-IO IX0,1 &
```



```
%{ #include <stdio.h> %}
if (IX0.0) { printf("hello, world\n"); }
```

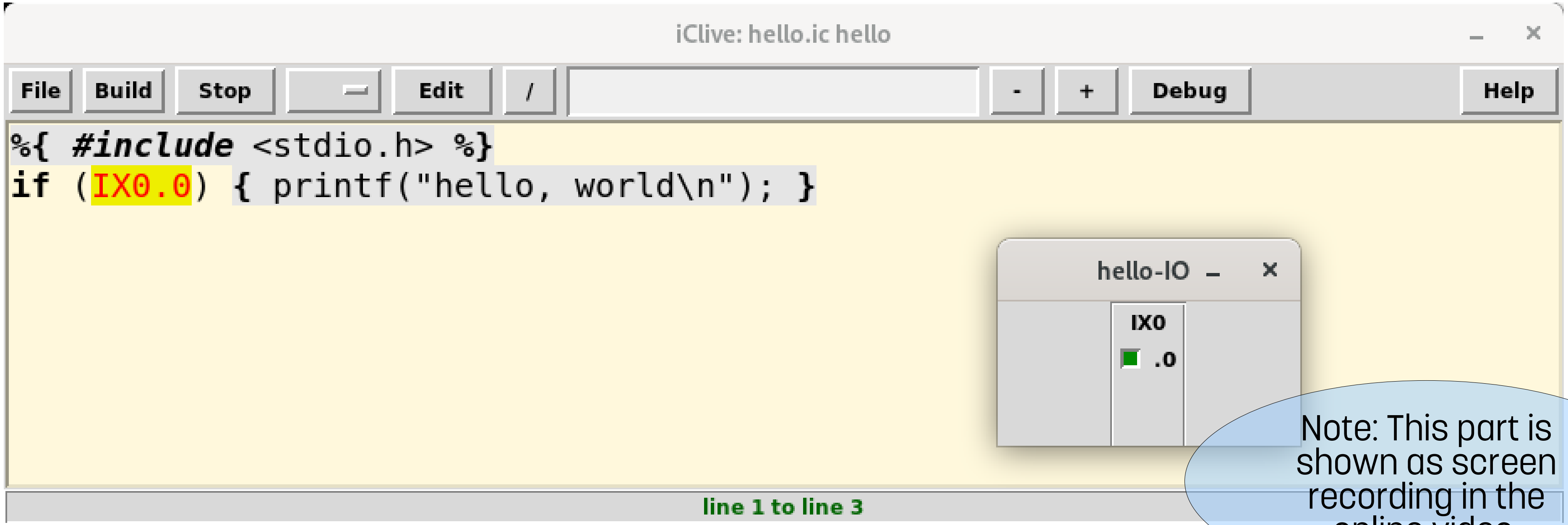
'hello' is running

Note: This part is shown as screen recording in the online video.

Testing the application: Live

Live, IX0.0 → High

```
$ hello, world
```



```
%{ #include <stdio.h> %}
if (IX0.0) { printf("hello, world\n"); }
```

Note: This part is shown as screen recording in the online video.

Summary

- OSADL provides a **license compliant container base image** in different variants.
- Customized applications can be built on top of this OSADL Docker Base Image.

- **Disclaimer:** The information given in this presentation does not constitute individual legal advice. It is based on a legal opinion (see references given on the last slide). There is as of yet no case law with regard to this topic, and different interpretations may certainly be arguable.

References

- Hemel, A., 2020, Docker Containers for Legal Professionals. [pdf] Available at: https://www.linuxfoundation.org/wp-content/uploads/Docker-Containers-for-Legal-Professionals-Whitepaper_042420.pdf [Accessed August 25, 2021]
- Jaeger, T., 2021, OSADL Legal Opinion: Are license obligations to be fulfilled when distributing Dockerfiles?