Docker & why we should use it

Vicențiu Ciorbaru
Software Engineer @ MariaDB Foundation

Agenda

- What is Docker?
- What Docker brings to the table compared to KVM and Vagrant?
- Docker tutorial

What is Docker

- A tool to manage containers (similar to Vagrant)
- Allows portable sharing of containers across machines
- Made use of LXC containers initially, now runC (libcontainer)
- Makes use of AuFS to limit size of containers

Why Docker?

- Much lower overhead than Virtual Machines
- Easy to share containers between devs
- Easy to run identical environment locally

How Docker Works

- 2 main concepts
  - images
  - containers

What is a docker image

- A set of files (binaries in general)
  - With an absolute path attached to each one
- That's it!
- Can be based off of other images
  - Tracked via AuFS

What is a docker container

- A docker image running 1 or more processes started from the binaries in the image.
- Any changes to files are recorded on a separate layer.

What is a Dockerfile

- A script to generate a docker image.
- Each instruction generates an intermediate image, based on modified files.
- Used by the command *docker build*.

What is a Dockerfile

```
# Base image
FROM ubuntu:16.04

# Create user develop
RUN useradd -m -b /home --uid 1000 develop

# Install build dependencies
RUN apt-get update \
     && apt-get install -y \
        git cmake gcc g++ \
        libncurses5-dev \
        libgnutls-dev \
        bison \
     && rm -rf /var/lib/apt/lists/*
```

```
# Copy the repository
ADD ./server /home/develop/server
RUN chown -R develop /home/develop
USER develop

# Build the server
WORKDIR /home/develop/server
RUN cmake . -DCMAKE_BUILD_TYPE=Debug
RUN make -j13

# Run tests when container is run
WORKDIR /home/develop/server/mysql-test
CMD ./mtr --par=20 --suite=main --force
```

How a docker image is constructed

```
# Base image
FROM ubuntu:16.04

# Create user develop
RUN useradd -m -b /home --uid 1000 develop

# Install build dependencies
RUN apt-get update \
     && apt-get install -y \
        git cmake gcc g++ \
        libncurses5-dev \
        libgnutls-dev \
        bison \
     && rm -rf /var/lib/apt/lists/*
```

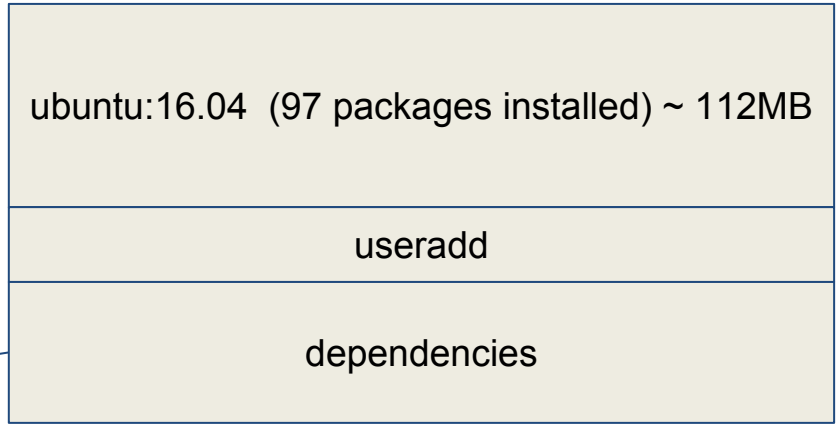
ubuntu:16.04 (97 packages installed) ~ 112MB

How a docker image is constructed

```
# Base image
FROM ubuntu:16.04

# Create user develop
RUN useradd -m -b /home --uid 1000 develop

# Install build dependencies
RUN apt-get update \
     && apt-get install -y \
        git cmake gcc g++ \
        libncurses5-dev \
        libgnutls-dev \
        bison \
     && rm -rf /var/lib/apt/lists/*
```

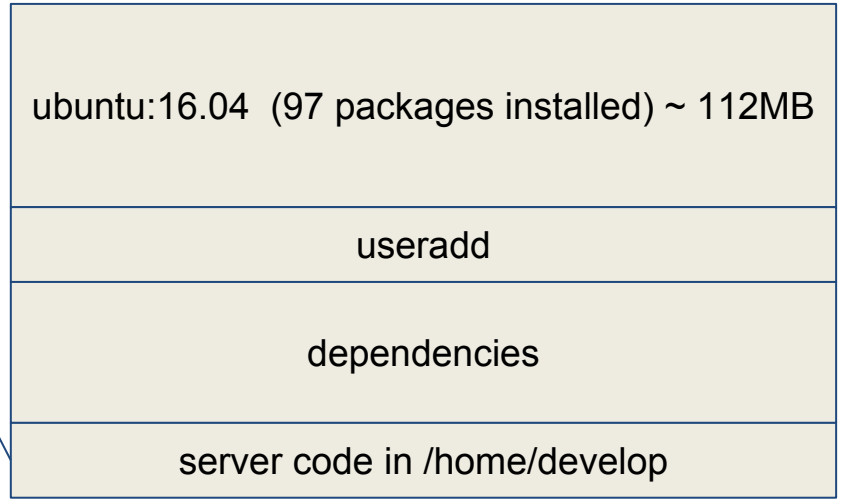
How a docker image is constructed

```
# Base image
FROM ubuntu:16.04

# Create user develop
RUN useradd -m -b /home --uid 1000 develop

# Install build dependencies
RUN apt-get update \
     && apt-get install -y \
        git cmake gcc g++ \
        libncurses5-dev \
        libgnutls-dev \
        bison \
     && rm -rf /var/lib/apt/lists/*
```



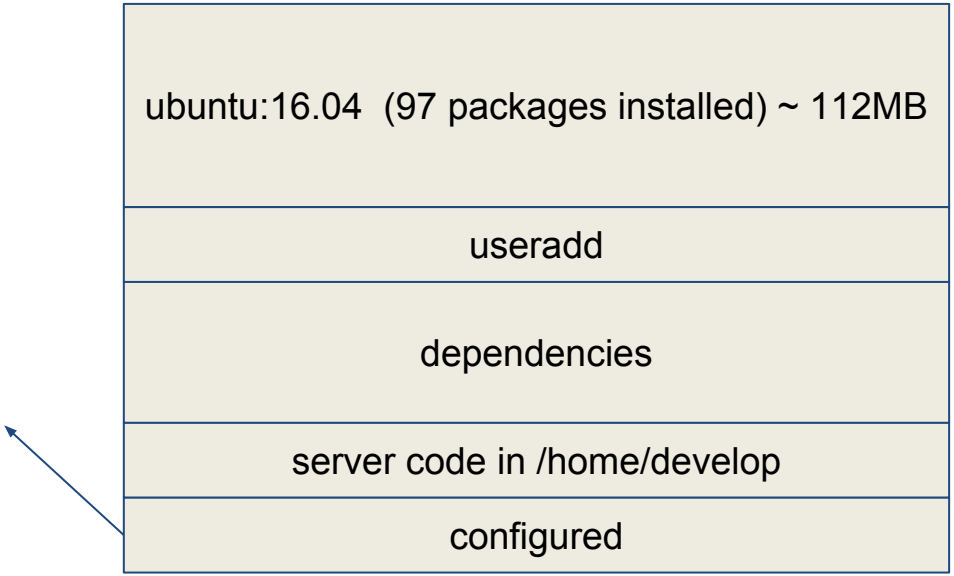
How a docker image is constructed

```
# Copy the repository
ADD ./server /home/develop/server
RUN chown -R develop /home/develop
USER develop

# Build the server
WORKDIR /home/develop/server
RUN cmake . -DCMAKE_BUILD_TYPE=Debug
RUN make -j13

# Run tests when container is run
WORKDIR /home/develop/server/mysql-test
CMD ./mtr --par=20 --suite=main --force
```



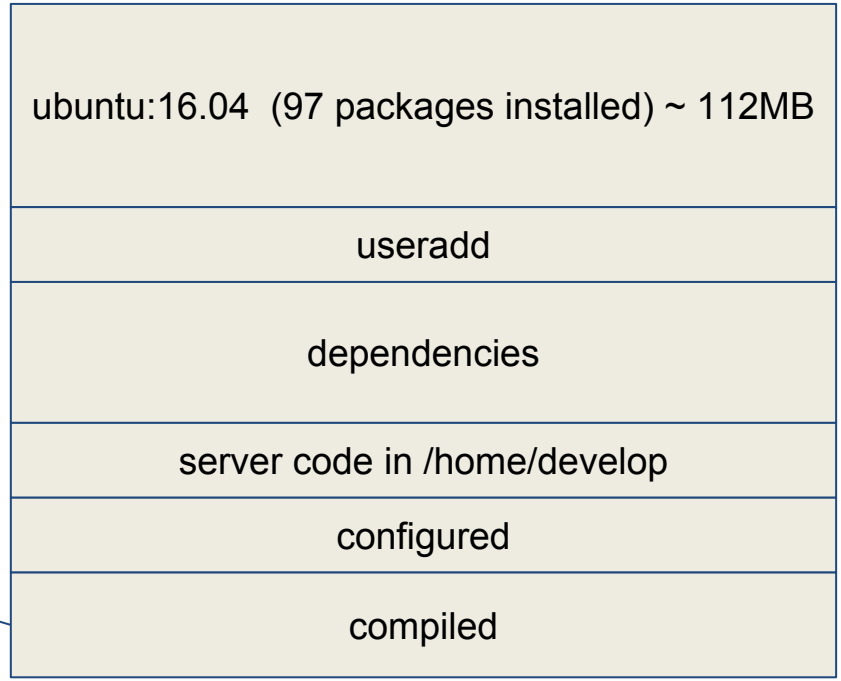
How a docker image is constructed

```
# Copy the repository
ADD ./server /home/develop/server
RUN chown -R develop /home/develop
USER develop

# Build the server
WORKDIR /home/develop/server
RUN cmake . -DCMAKE_BUILD_TYPE=Debug
RUN make -j13

# Run tests when container is run
WORKDIR /home/develop/server/mysql-test
CMD ./mtr --par=20 --suite=main --force
```



How a docker image is constructed

```
# Copy the repository
ADD ./server /home/develop/server
RUN chown -R develop /home/develop
USER develop

# Build the server
WORKDIR /home/develop/server
RUN cmake . -DCMAKE_BUILD_TYPE=Debug
RUN make -j13

# Run tests when container is run
WORKDIR /home/develop/server/mysql-test
CMD ./mtr --par=20 --suite=main --force
```



Installing Docker

```
$ sudo apt install docker.io
$ sudo systemctl enable docker.io
$ sudo systemctl start docker.io
$ docker ps    # By default, only root can run docker commands
Got permission denied while trying to connect to the Docker daemon socket
$ sudo groupadd docker  # May already exist
$ sudo usermod -a -G docker $USER
$ docker ps
CONTAINER ID    IMAGE     COMMAND    CREATED    STATUS    PORTS     NAMES
```

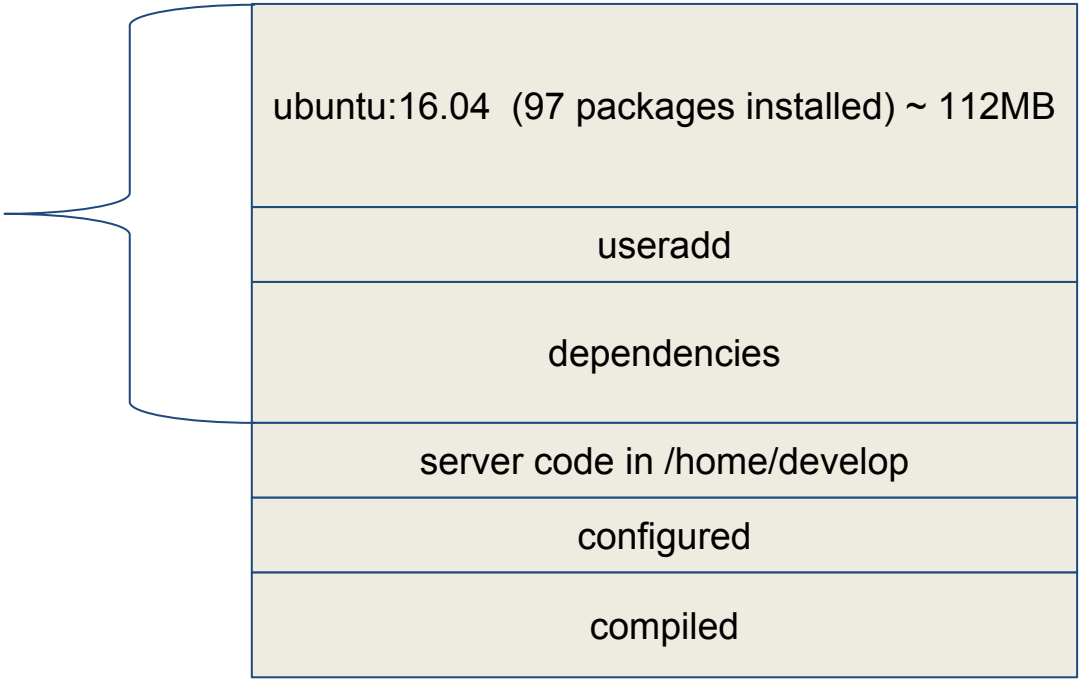
Running MariaDB tests in Docker

```
$ mkdir ~/Workspace/MariaDB-docker

$ cd ~/Workspace/MariaDB-docker

$ vim Dockerfile  # Use the dockerfile from previous slides

$ docker build . -t mariadb-test

$ docker run -i -t mariadb-test
```

How to get reproducible builds

Build Environment

```
152M build-env.tgz
```

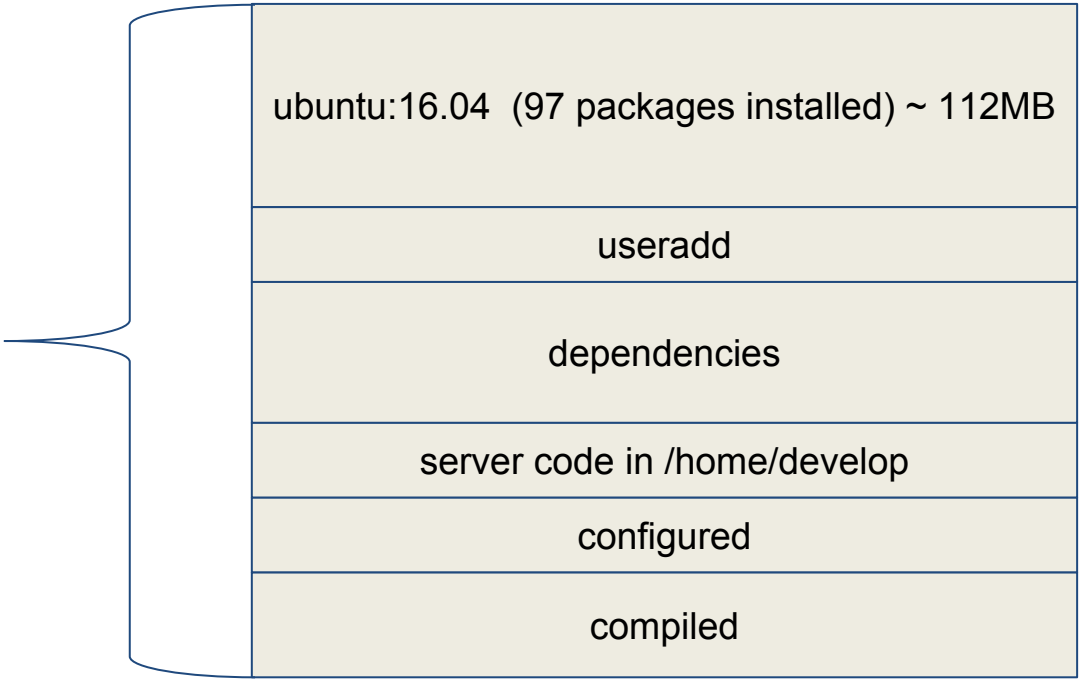


How to get reproducible builds

Test Environment

3.1GB test-env.tgz

:(Too big, code and binaries take too much space



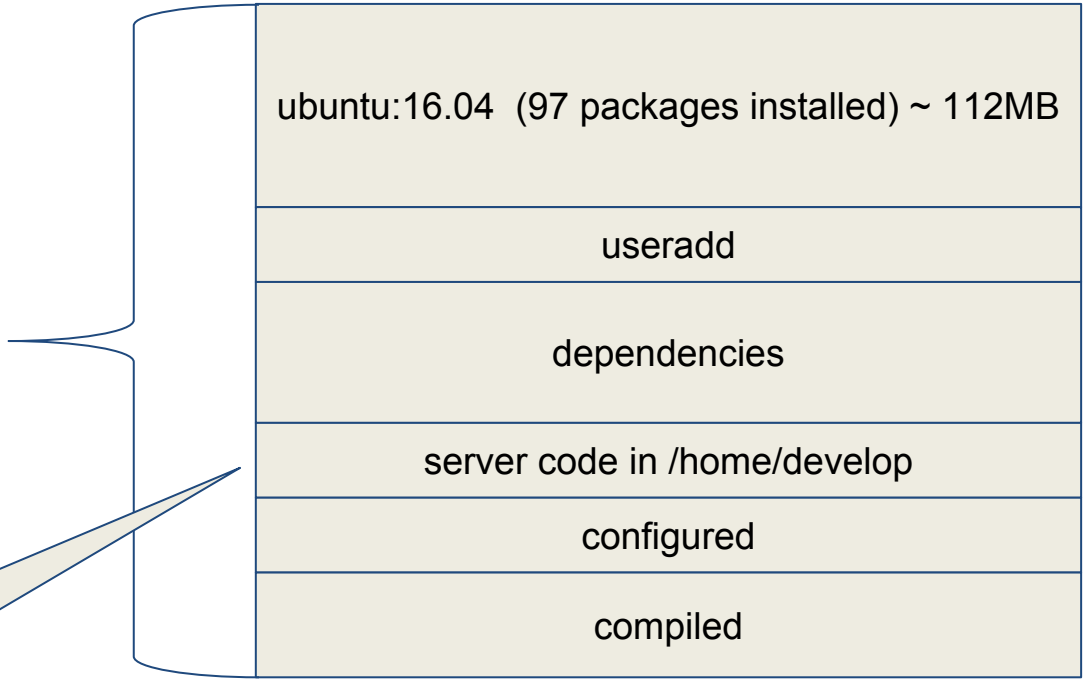
How to get reproducible builds

Test Environment

3.1GB test-env.tgz

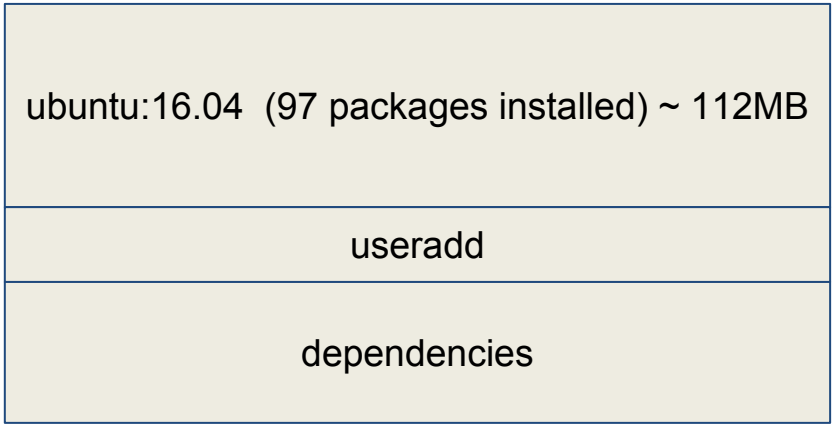
:(Too big, code and binaries take too much space

We need to remove the code from the image.



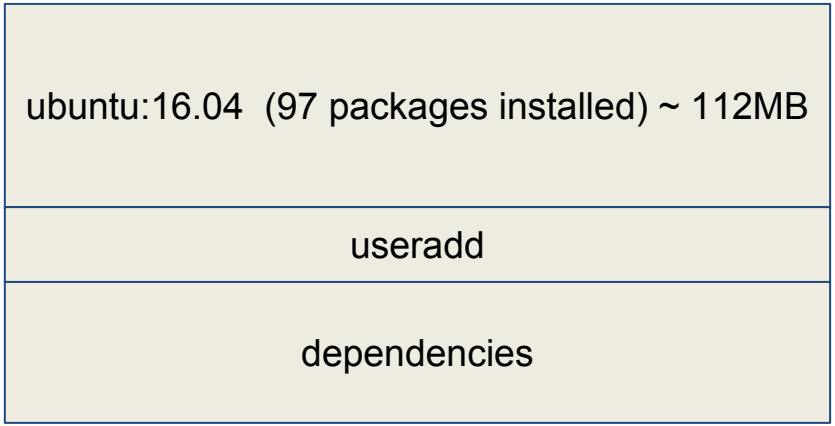
Keep images small, use volumes

```
# Base image
FROM ubuntu:16.04

# Create user develop
RUN useradd -b /home --uid 1000 develop

# Install build dependencies
RUN apt-get update \
     && apt-get install -y \
        git cmake gcc g++ \
        libncurses5-dev \
        libgnutls-dev \
        bison \
     && rm -rf /var/lib/apt/lists/*
USER develop
```



Keep images small, use volumes

```
$ docker build . -t mariadb-build-env
```

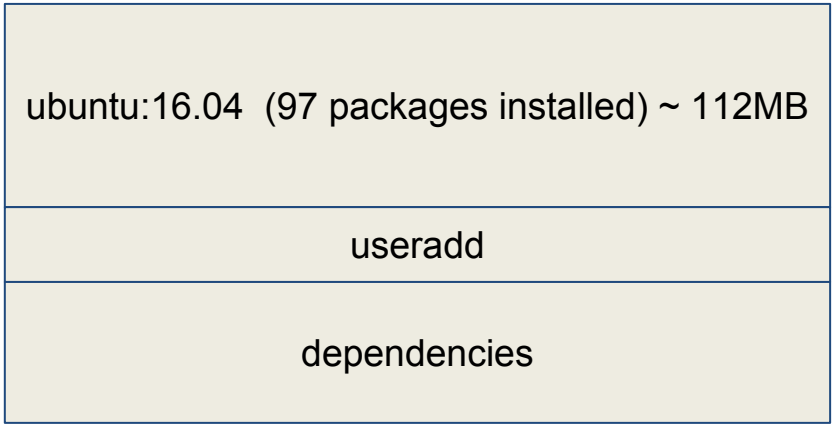



```
$ docker run -i -t -v "$(pwd)"/server:/home/develop/server  mariadb-build-env
```

Specify a volume to mount

Keep images small, use volumes

```
$ docker build . -t mariadb-build-env
```

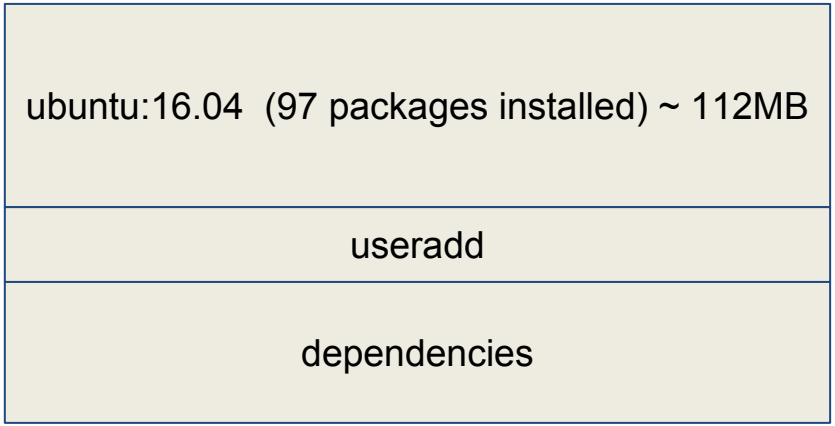



```
$ docker run -i -t -v "$(pwd)"/server:/home/develop/server  mariadb-build-env
```

Absolute path in host

Keep images small, use volumes

```
$ docker build . -t mariadb-build-env
```

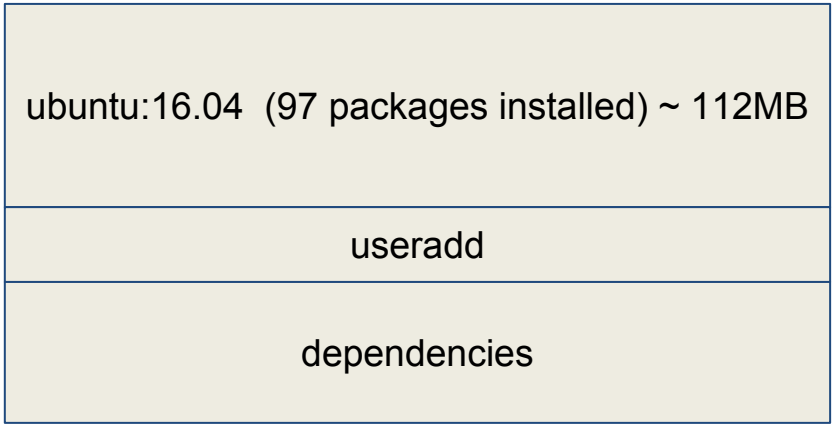



```
$ docker run -i -t -v "$(pwd)"/server:/home/develop/server  mariadb-build-env
```

Absolute path in container

Keep images small, use volumes

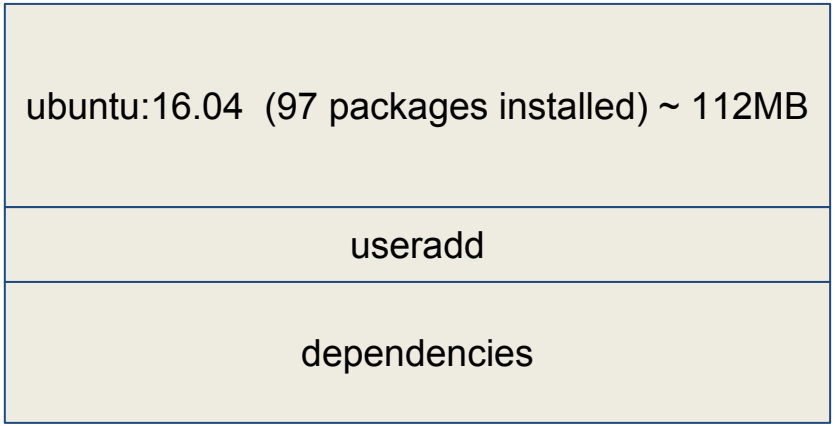
```
$ docker build . -t mariadb-build-env
```




```
$ docker run -i -t -v "$(pwd)"/server:/home/develop/server  mariadb-build-env

develop@b624906b23a5:~$
```

Keep images small, use volumes

```
$ docker build . -t mariadb-build-env
```




```
$ docker run -i -t -v "$(pwd)"/server:/home/develop/server  mariadb-build-env

develop@b624906b23a5:~$ ls
server
develop@b624906b23a5:~$ cd server
develop@b624906b23a5:~/server$ cmake .
-- Running cmake version 3.5.1
...
```

Conclusions

- Docker can be used to test and debug any specific environment, quickly
- Sharing images can be done in 2 ways:
  - Share the Dockerfile (fast, easy)
  - Publish the image to a registry, share the image name (similar to git push)
- To reduce image size, use volumes for large contents

Conclusions

- Developing in Docker containers should (mostly) remove the "can not reproduce" problem!
- Guaranteed identical environment every time
- Everything runs in regular userspace, no hypervizor overhead!
- There are more customization features present:
  - networks, ports mappings, docker-compose

Thank You!

Contact me at:
vicentiu@mariadb.org
vicentiu@ciorbaru.io

Blogs:
mariadb.org/blog
vicentiu.ciorbaru.io