

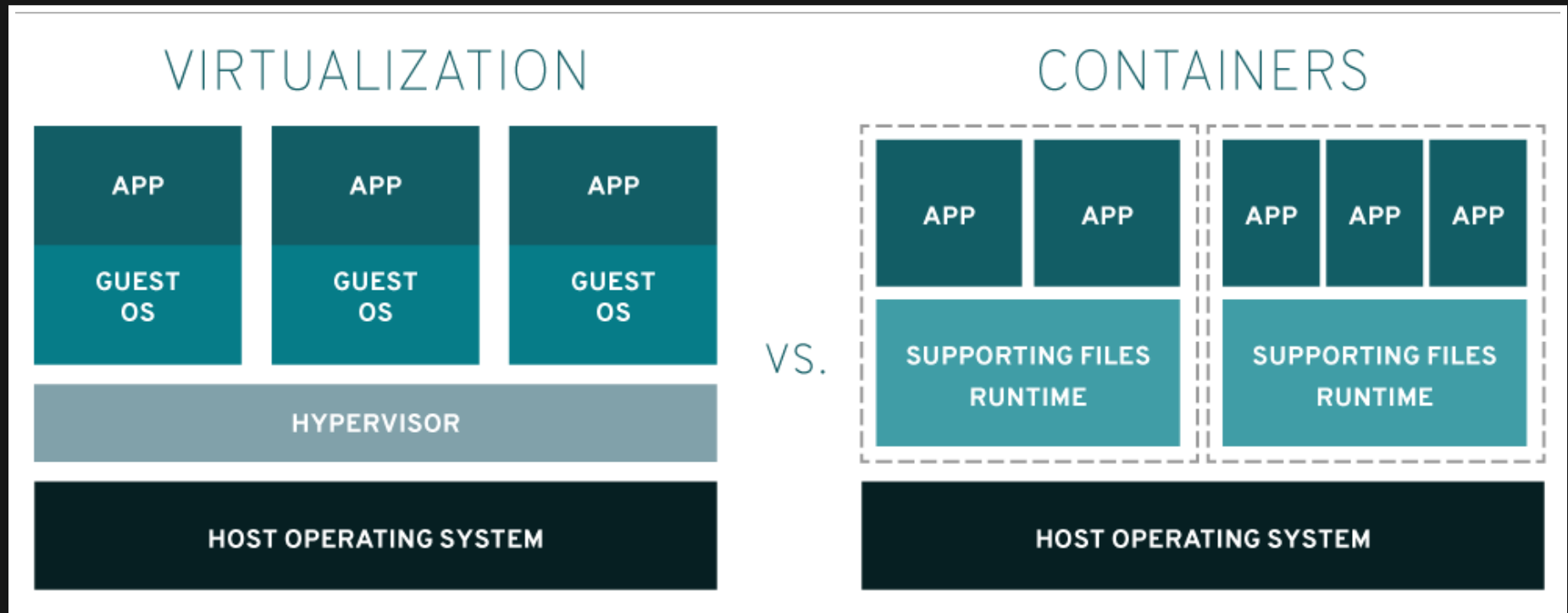
# WORKING WITH DOCKER



Enrico Nasca  
Research engineer at IKIM

7 June 2021

Containers provide a fairly good balance between virtualisation and native execution.



Source: [Red Hat](#)

A container includes OS components — except the kernel — and one or more applications.

A container includes OS components — except the kernel — and one or more applications.

- What if you run x86\_64 Linux containers on an x86\_64 Linux host?

A container includes OS components — except the kernel — and one or more applications.

- What if you run x86\_64 Linux containers on an x86\_64 Linux host?  
They all share the kernel with the host.

A container includes OS components — except the kernel — and one or more applications.

- What if you run x86\_64 Linux containers on an x86\_64 Linux host?  
They all share the kernel with the host.
- What about a Linux container on a different host OS?

A container includes OS components — except the kernel — and one or more applications.

- What if you run x86\_64 Linux containers on an x86\_64 Linux host?  
They all share the kernel with the host.
- What about a Linux container on a different host OS?  
There is no Linux kernel, therefore Docker uses a virtual machine.

A container includes OS components — except the kernel — and one or more applications.

- What if you run x86\_64 Linux containers on an x86\_64 Linux host?  
They all share the kernel with the host.
- What about a Linux container on a different host OS?  
There is no Linux kernel, therefore Docker uses a virtual machine.
- What about an x86\_64 Linux container on an arm64v8 Linux host?



A container includes OS components — except the kernel — and one or more applications.

- What if you run x86\_64 Linux containers on an x86\_64 Linux host?  
They all share the kernel with the host.
- What about a Linux container on a different host OS?  
There is no Linux kernel, therefore Docker uses a virtual machine.
- What about an x86\_64 Linux container on an arm64v8 Linux host?  
Incompatible architectures: Docker adds a VM to the mix.

A container includes OS components — except the kernel — and one or more applications.

- What if you run x86\_64 Linux containers on an x86\_64 Linux host?  
They all share the kernel with the host.
- What about a Linux container on a different host OS?  
There is no Linux kernel, therefore Docker uses a virtual machine.
- What about an x86\_64 Linux container on an arm64v8 Linux host?  
Incompatible architectures: Docker adds a VM to the mix.
- Can containers use devices such as GPUs?

A container includes OS components — except the kernel — and one or more applications.

- What if you run x86\_64 Linux containers on an x86\_64 Linux host?  
They all share the kernel with the host.
- What about a Linux container on a different host OS?  
There is no Linux kernel, therefore Docker uses a virtual machine.
- What about an x86\_64 Linux container on an arm64v8 Linux host?  
Incompatible architectures: Docker adds a VM to the mix.
- Can containers use devices such as GPUs?  
Yes, you just need to expose the device to the container and install the appropriate OS components in the container.

# COMMAND-LINE DOCKER

```
# Download an image
docker pull alpine

# Create a container, run a command and stop immediately.
docker run --name=helloalpine1 alpine echo hello

# Create a container and keep it running.
docker run --name=helloalpine2 alpine tail -f /dev/null

# To the running container, open another shell and execute:
docker stop helloalpine2

# Clean up
docker rm helloalpine1 helloalpine2
```

Lifecycle of a container:  
create → start → stop → remove.

Changes made in a container are preserved when it's stopped, but are lost when it's removed.

1. `docker run -it --name=helloalpine alpine`

2. Make a change, such as `touch hello.txt`, then exit.

3. `docker start -i helloalpine`

- The file is still present.

4. `docker rm helloalpine`

- All files are gone, but the image is still on storage and can be used to create new containers.

Filesystem mounts allow sharing persistent storage with the container.

```
justme@home:~$ mkdir testdir
justme@home:~$ docker run --rm -it \
  --mount type=bind,src="$PWD"/testdir,dst=/data \
  alpine
```

The path `./testdir` on the host is now bound-mounted to `/data` in the container. Changes to one location are reflected in the other.

# What happens with file ownership?

This is better demonstrated on a Linux host, such as an IKIM node.

```
justme@home:~$ ssh \
  -J justme@login.ikim.uk-essen.de \
  justme@g1-2

justme@g1-2:~$ mkdir testdir
justme@g1-2:~$ docker run --rm -it \
  --mount type=bind,src="$PWD"/testdir,dst=/data \
  alpine
/ # echo hello > /data/hello.txt

justme@g1-2:~$ ls -l testdir/hello.txt
-rw-r--r-- 1 root root 6 Jun  5 08:44 testdir/hello.txt
```

# The file is owned by root, even if you're neither root nor have passwordless sudo. Why?

```
justme@g1-2:~$ groups  
justme docker
```

```
justme@g1-2:~$ ls -l /var/run/docker.sock  
srw-rw---- 1 root docker 0 Mar 31 13:14 /var/run/docker.sock
```

- You belong to the `docker` group, which can write to the Docker socket.
- By writing to the Docker socket, you send commands to Docker Engine, which runs as root.
- As a result, you are effectively root. You can do scary things such as bind-mounting a path where you couldn't normally write.



# DOCKERFILES

Docker images can be created by building on top of existing images.

Create a file called `Dockerfile` in the current directory.

```
FROM ubuntu
ARG USERNAME=justme
ARG USER_UID=1000
ARG USER_GID=$USER_UID
RUN groupadd --gid $USER_GID $USERNAME \
    && useradd --uid $USER_UID --gid $USER_GID -m $USERNAME
USER $USERNAME
```

Build and run the image.

```
justme@g1-2:~$ docker build -t ubuntu-nonroot .
justme@g1-2:~$ docker run --rm -it ubuntu-nonroot
```

# PORTABILITY

- The root user in the container is root outside of the container too. The same goes for all other UIDs and GIDs.
- If you want to be "you" in a container, build an image and create an account with the same UID+GIDs.

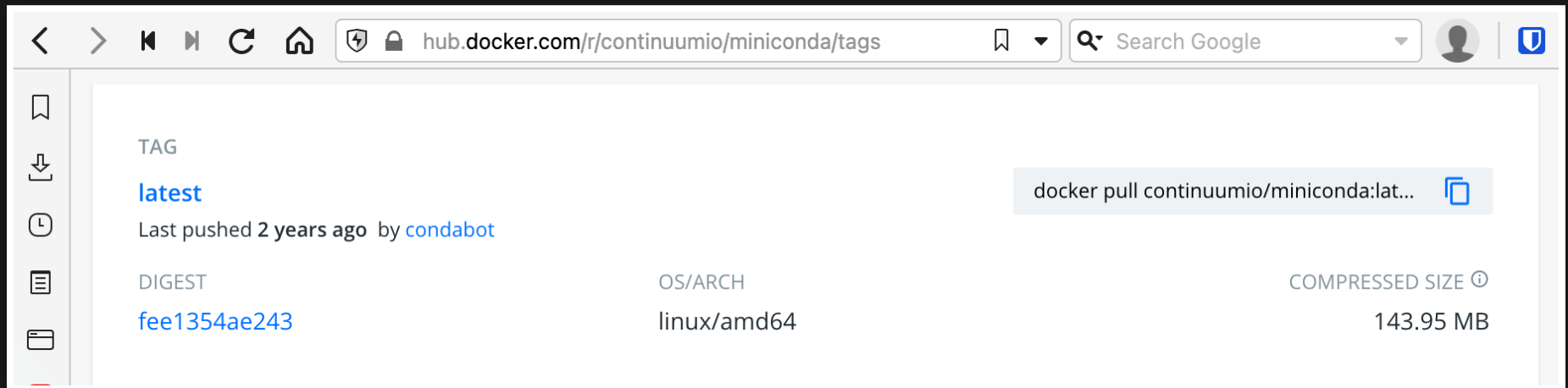
How to find the relevant information?

```
justme@g1-2:~$ id  
uid=1014(justme) gid=1014(justme) groups=1014(justme),998(dock
```

These considerations apply when Docker can use the native kernel.

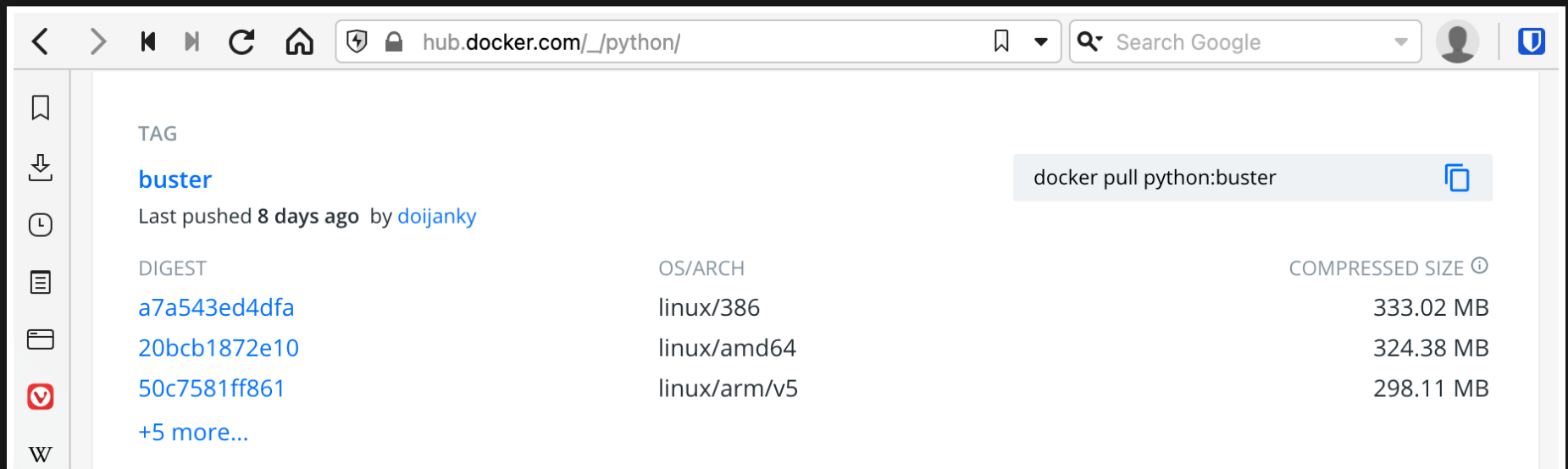
In other cases, such as with a macOS host, the VM layer handles users and privileges.

# Pay attention to the CPU architecture of a Docker image.



A screenshot of a web browser showing the Docker Hub page for the `continuumio/miniconda` image. The browser's address bar shows `hub.docker.com/r/continuumio/miniconda/tags`. The page displays the `latest` tag, pushed 2 years ago by `condabot`. A table lists the image's details: Digest `fee1354ae243`, OS/ARCH `linux/amd64`, and Compressed Size `143.95 MB`. A pull command `docker pull continuumio/miniconda:lat...` is shown in a box.

| TAG  |                          |
|--|--------------------------|
| <b>latest</b>                                    |                          |
| Last pushed 2 years ago by <code>condabot</code> |                          |
| DIGEST   | OS/ARCH                  |
| <code>fee1354ae243</code>                        | <code>linux/amd64</code> |
| COMPRESSED SIZE ⓘ                                |                          |
| <code>143.95 MB</code>                           |                          |



A screenshot of a web browser showing the Docker Hub page for the `python` image. The browser's address bar shows `hub.docker.com/_/python/`. The page displays the `buster` tag, pushed 8 days ago by `doijanky`. A table lists the image's details for different architectures: `linux/386` (Digest `a7a543ed4dfa`, Size `333.02 MB`), `linux/amd64` (Digest `20bcb1872e10`, Size `324.38 MB`), and `linux/arm/v5` (Digest `50c7581ff861`, Size `298.11 MB`). A pull command `docker pull python:buster` is shown in a box.

| TAG   |                           |
|---|---------------------------|
| <b>buster</b>                                   |                           |
| Last pushed 8 days ago by <code>doijanky</code> |                           |
| DIGEST  | OS/ARCH                   |
| <code>a7a543ed4dfa</code>                       | <code>linux/386</code>    |
| <code>20bcb1872e10</code>                       | <code>linux/amd64</code>  |
| <code>50c7581ff861</code>                       | <code>linux/arm/v5</code> |
| <a href="#">+5 more...</a>                      |                           |
| COMPRESSED SIZE ⓘ                               |                           |
| <code>333.02 MB</code>                          |                           |
| <code>324.38 MB</code>                          |                           |
| <code>298.11 MB</code>                          |                           |

# THANKS!

Further reading:

- [Docker Compose](#):  
multi-container applications.
- [Devcontainers in Visual Studio Code](#):  
portable (with the usual caveats), self-documenting  
development environments.
- Container orchestration with [Kubernetes](#):  
deploy containerised services in a cluster.