SESH & CompSoc

Docker

The Legal Bit

- The skills taught in these sessions allow identification and exploitation of security vulnerabilities in systems. We strive to give you a place to practice legally, and can point you to other places to practice. These skills should not be used on systems where you do not have explicit permission from the owner of the system. It is VERY easy to end up in breach of relevant laws, and we can accept no responsibility for anything you do with the skills learnt here.

- If we have reason to believe that you are utilising these skills against systems where you are not authorised you will be banned from our events, and if necessary the relevant authorities will be alerted.

- Remember, if you have any doubts as to if something is legal or authorised, just don't do it until you are able to confirm you are allowed to.

- Relevant UK Law: https://www.legislation.gov.uk/ukpga/1990/18/contents

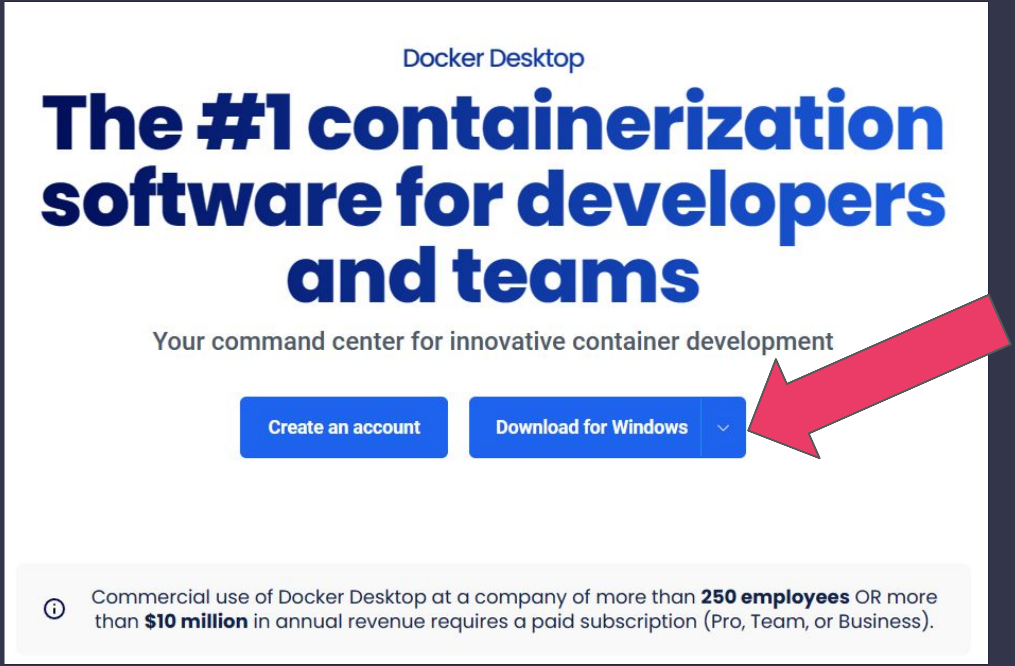
Code of Conduct

- Before proceeding past this point you must read and agree to our Code of Conduct - this is a requirement from the University for us to operate as a society.
- If you have any doubts or need anything clarified, please ask a member of the committee.
- Breaching the Code of Conduct = immediate ejection and further consequences.
- Code of Conduct can be found at https://shefesh.com/conduct

Downloading Docker Desktop

Later on in this session, we will be using software called Docker Desktop which can be downloaded on Windows, Mac and Linux here: https://www.docker.com/products/docker-desktop/

Downloading docker via CLI: https://docs.docker.com/engine/install/ubuntu/



Docker Desktop

The #1 containerization software for developers and teams

Your command center for innovative container development

Commercial use of Docker Desktop at a company of more than **250 employees** OR more than **$10 million** in annual revenue requires a paid subscription (Pro, Team, or Business).

Hover over the arrow for OS options

What is docker

- Docker is a containerisation platform that packages an application and its dependencies together inside of an image.
- Docker enables you to separate applications from infrastructure to increase the efficiency of delivering software.
- A container provides a self-contained environment for running applications and software.
- Containers are isolated from one another and underlying infrastructure so they can run in any environment.
- Containers are alternatives to using virtual machines as it uses fewer resources and runs on top of the host’s OS.

Why you should use docker

- Provides portability across different machines as you can deploy containers to any other machine that runs Docker
- More efficient than virtual machines as they do not contain an OS
- Development process of applications and software is more fluid
- Docker containers allow for faster delivery of software updates and rollbacks
- Can also repair applications without completely taking it down
- Containers provide an isolated environment so all required resources self contained to prevent disturbing or depending on another container
- Easy to install and use

Companies that use docker:

- Paypal
- Adobe

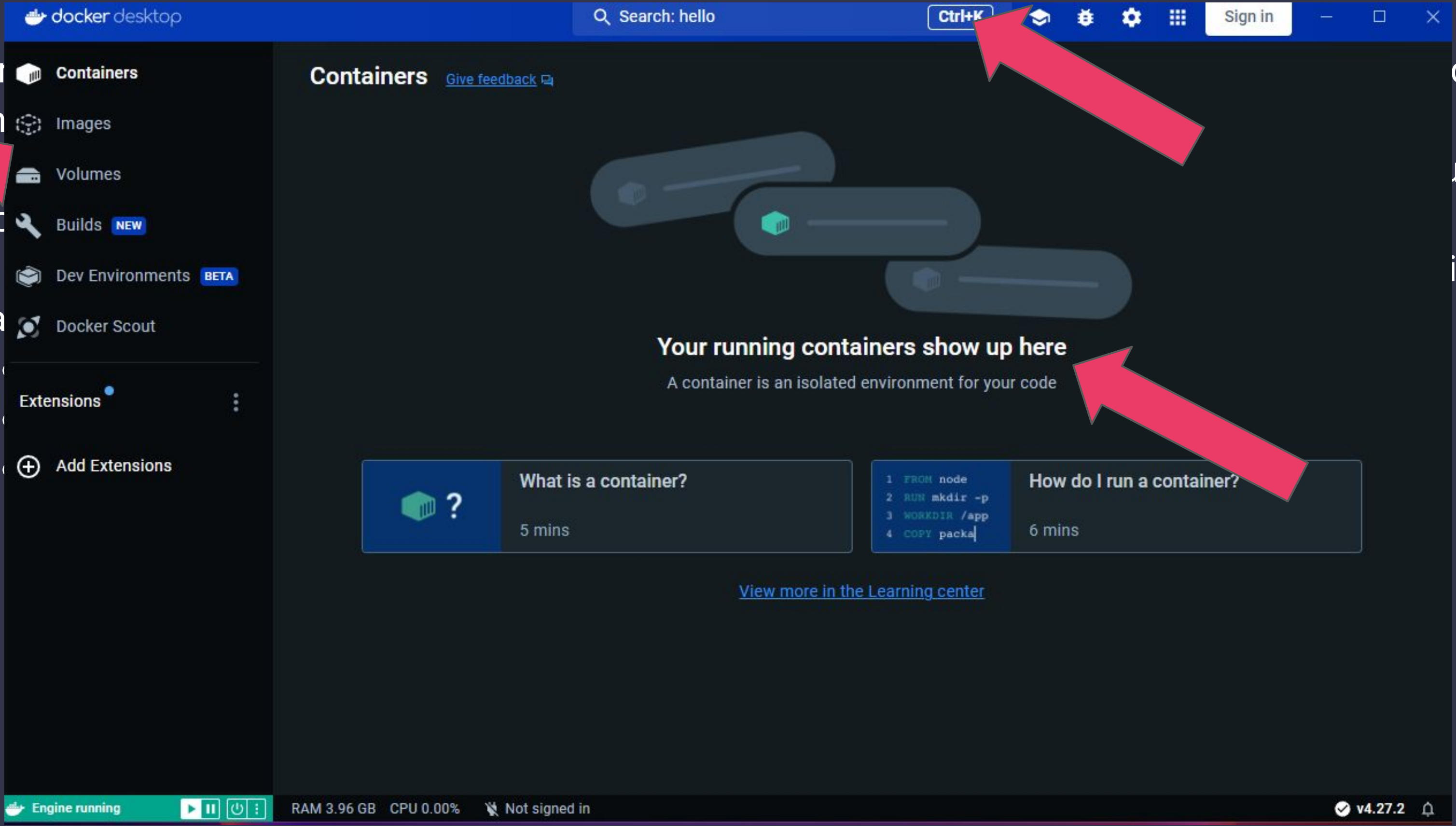
Where you can use docker

You can use Docker on any machine that can run docker for a range of things such as:

- Software prototyping and packaging
- Network modelling
- Continuous integration and delivery
- Running multiple containers on the same machine
- Databases (you can keep data by binding Docker to a volume which will be discussed later)
- Early application development
- Pre-deployment testing

Docker desktop

- Pr ds to th
- up so yo
- A iner (a



docker desktop

Search: hello Ctrl+K

Sign in

- Containers
- Images
- Volumes
- Builds NEW
- Dev Environments BETA
- Docker Scout

Extensions

Add Extensions

Containers Give feedback

Your running containers show up here

A container is an isolated environment for your code

What is a container?

5 mins

```
FROM node
RUN mkdir -p
WORKDIR /app
COPY packa
```

How do I run a container?

6 mins

View more in the Learning center

Engine running

RAM 3.96 GB CPU 0.00% Not signed in

v4.27.2

Docker Command Line

Benefits of using Docker CLI:

- Full control of containers
- Allows for more in-depth customisation of configuration for containers/images
- Allows users to automate container management tasks
- Easier to integrate with automaton tools
- Manage Docker containers from your IDE of choice
- Has third party add-ons such as Fig.io

Disadvantages:

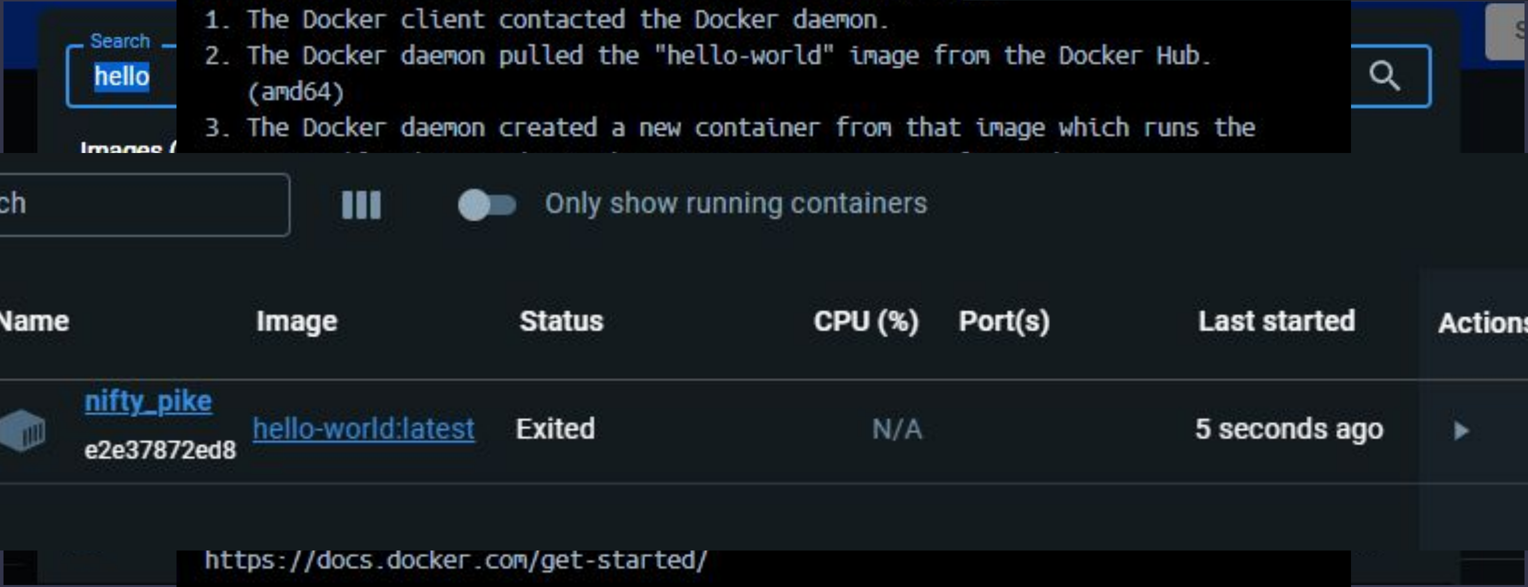
- Potential for security risks if not used properly
- Can be harder to debug
- Limited visual feedback for applications that need it

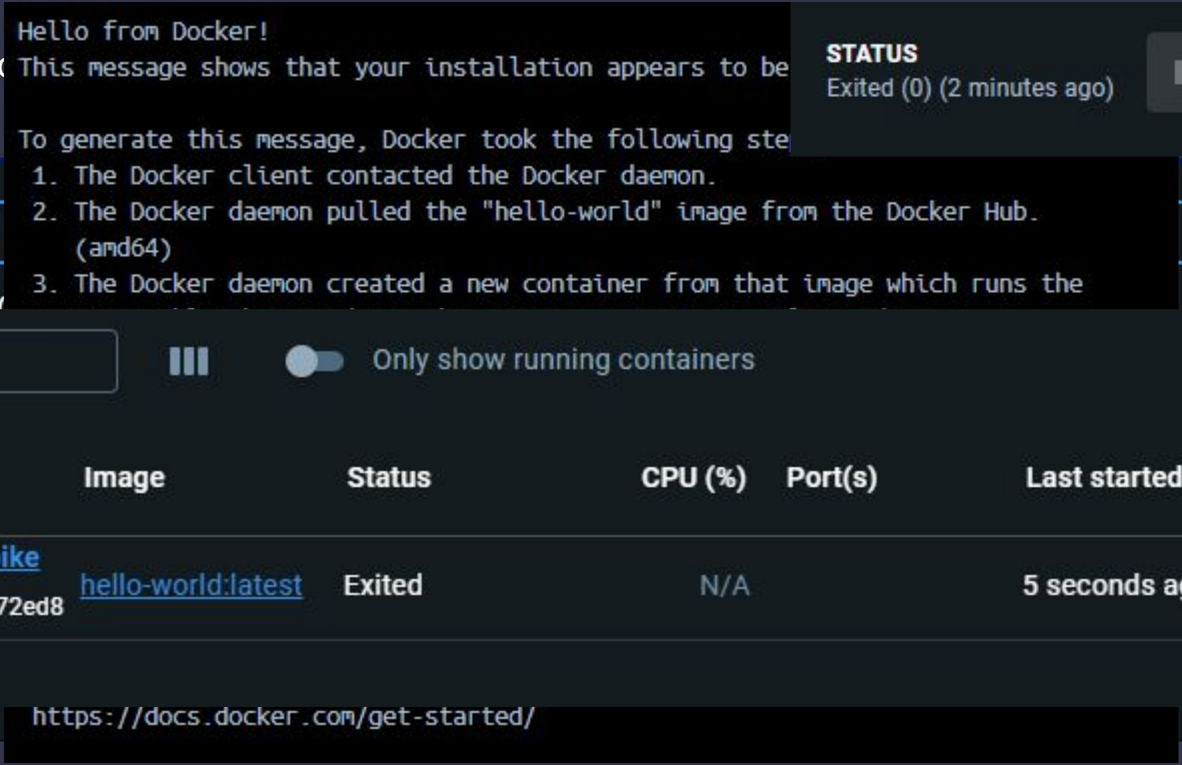
We will show certain CLI commands like this

hello-world

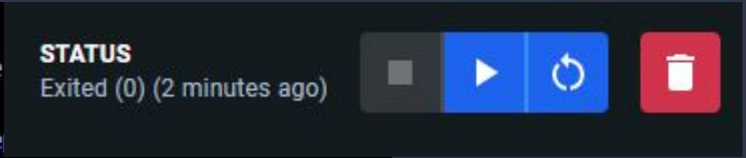
For CLI:
sudo docker run
hello-world

The hello world c

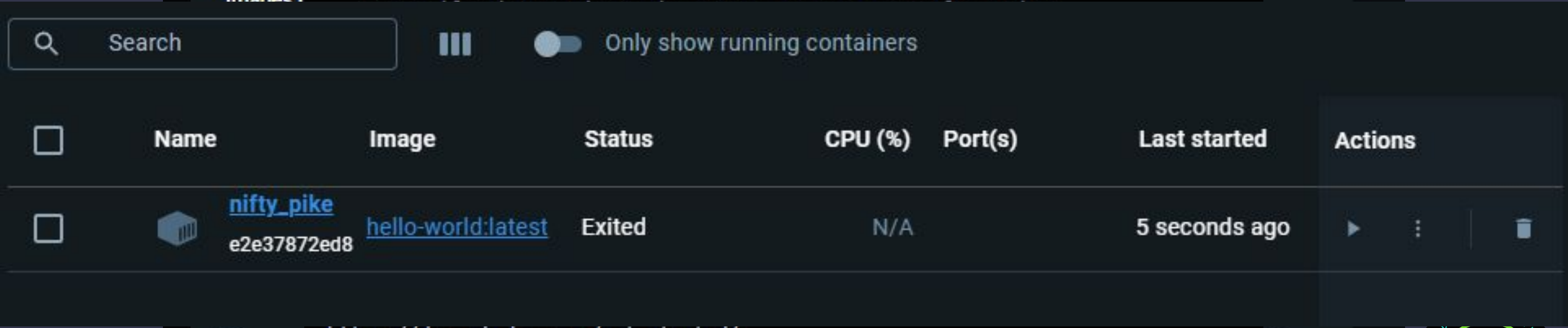




```
Hello from Docker!
This message shows that your installation appears to be

To generate this message, Docker took the following ste
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the

 https://docs.docker.com/get-started/
```



STATUS
Exited (0) (2 minutes ago)



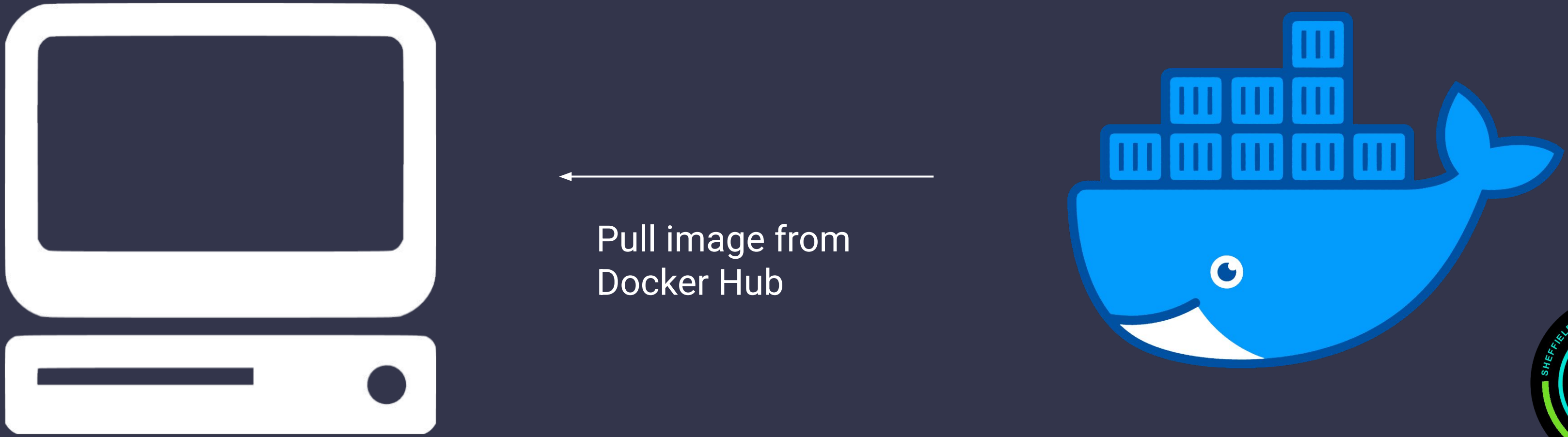
	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
☐	nifty_pike e2e37872ed8	hello-world:latest	Exited	N/A		5 seconds ago	

Pulling and Registries and Caches

```
docker login [DOCKER-REGISTRY-SERVER] -u <username> [-p <password>]
```

Pulling Images

- Images for containers can be large (those doing the IoT module will know)
- Images are pulled from a hub/registry and cached on the device
  - Many images are based of others which saves time if already downloaded
- Docker hub is main one for docker but you can host private hubs
  - Hubs and caches require configuration and upkeep



Images

Images are standardised packages that contain all the files, libraries and configurations needed to run a container

Once built, Images are Read-Only - but can be shared, versioned, and deployed as containers.

Can be pushed to docker hub, allowing others to pull them and use them

Bit like Github repositories - other users can pull and use apps created by someone else

- Docker pull <container-name>
- I.e Docker pull python3

Layers and SBOM (Desktop)

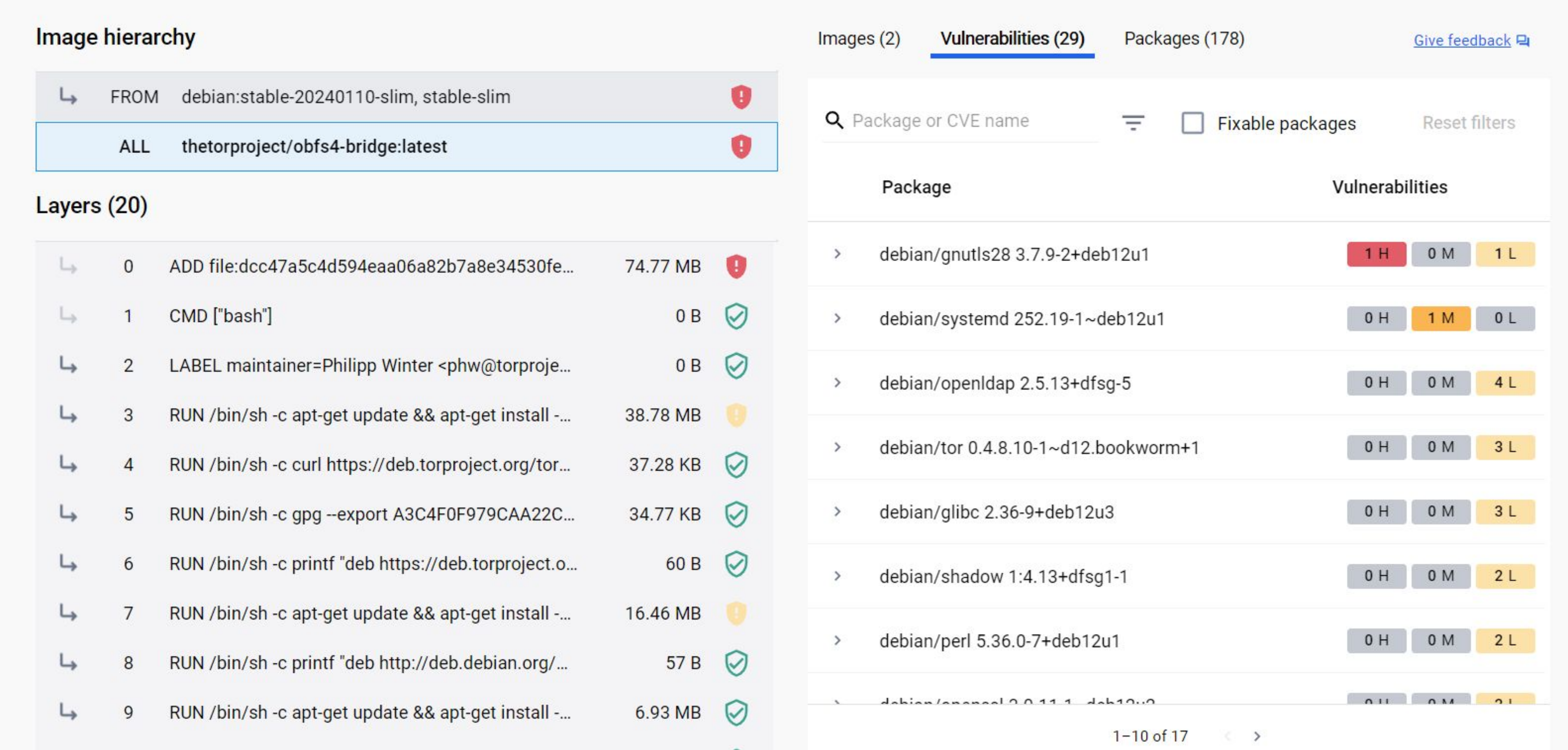


Image hierarchy

↳	FROM	debian:stable-20240110-slim, stable-slim	!
	ALL	thetorproject/obfs4-bridge:latest	!

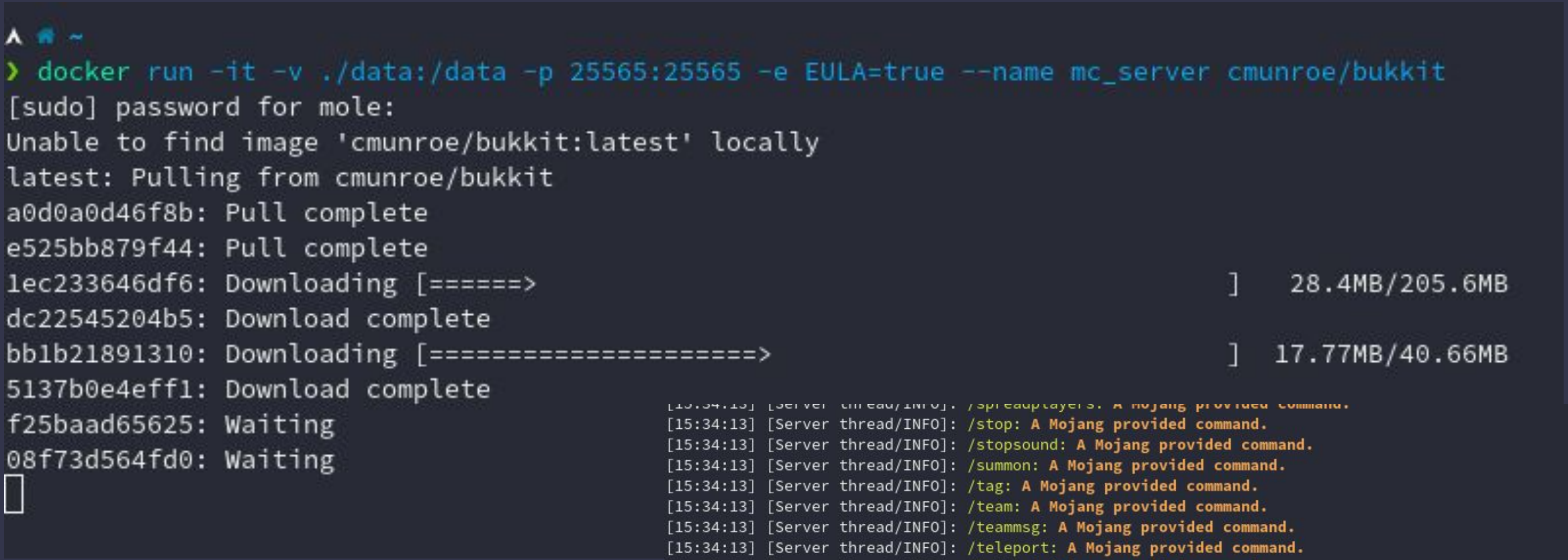
Layers (20)

↳	0	ADD file:dcc47a5c4d594eaa06a82b7a8e34530fe...	74.77 MB	!
↳	1	CMD ["bash"]	0 B	✓
↳	2	LABEL maintainer=Philipp Winter <phw@torproje...	0 B	✓
↳	3	RUN /bin/sh -c apt-get update && apt-get install -...	38.78 MB	!
↳	4	RUN /bin/sh -c curl https://deb.torproject.org/tor...	37.28 KB	✓
↳	5	RUN /bin/sh -c gpg --export A3C4F0F979CAA22C...	34.77 KB	✓
↳	6	RUN /bin/sh -c printf "deb https://deb.torproject.o...	60 B	✓
↳	7	RUN /bin/sh -c apt-get update && apt-get install -...	16.46 MB	!
↳	8	RUN /bin/sh -c printf "deb http://deb.debian.org/...	57 B	✓
↳	9	RUN /bin/sh -c apt-get update && apt-get install -...	6.93 MB	✓

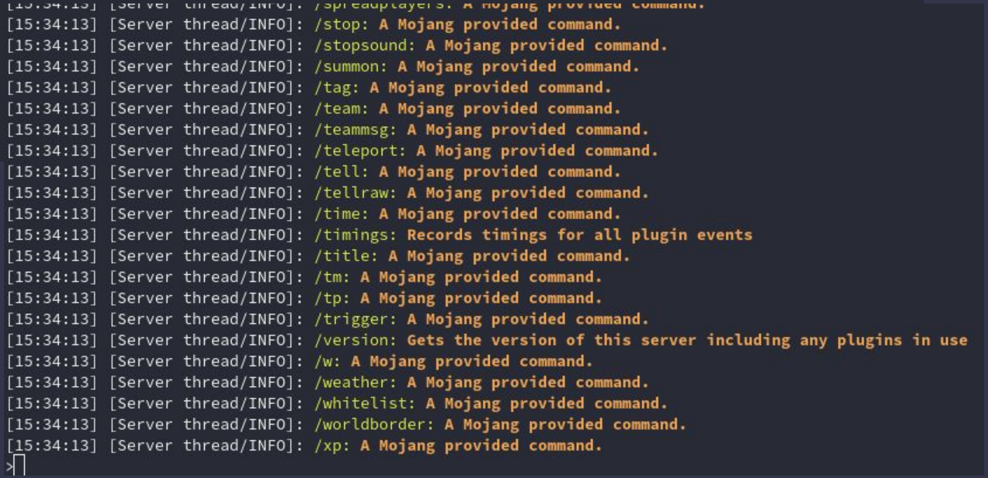
Package	Vulnerabilities		
debian/gnutls28 3.7.9-2+deb12u1	1 H	0 M	1 L
debian/systemd 252.19-1~deb12u1	0 H	1 M	0 L
debian/openldap 2.5.13+dfsg-5	0 H	0 M	4 L
debian/tor 0.4.8.10-1~d12.bookworm+1	0 H	0 M	3 L
debian/glibc 2.36-9+deb12u3	0 H	0 M	3 L
debian/shadow 1:4.13+dfsg1-1	0 H	0 M	2 L
debian/perl 5.36.0-7+deb12u1	0 H	0 M	2 L

1–10 of 17

Layers (CLI)



```
~
❯ docker run -it -v ./data:/data -p 25565:25565 -e EULA=true --name mc_server cmunroe/bukkit
[sudo] password for mole:
Unable to find image 'cmunroe/bukkit:latest' locally
latest: Pulling from cmunroe/bukkit
a0d0a0d46f8b: Pull complete
e525bb879f44: Pull complete
1ec233646df6: Downloading [======>                                            ]   28.4MB/205.6MB
dc22545204b5: Download complete
bb1b21891310: Downloading [=====================>                             ]  17.77MB/40.66MB
5137b0e4eff1: Download complete
f25baad65625: Waiting
08f73d564fd0: Waiting
```

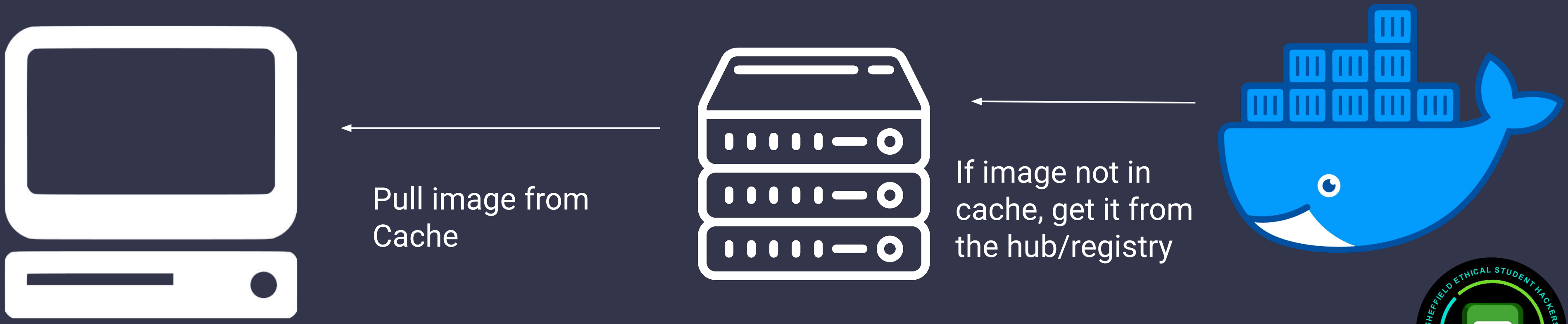


```
[15:34:13] [Server thread/INFO]: /spreadplayers: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /stop: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /stopsound: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /summon: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /tag: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /team: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /teammsg: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /teleport: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /tell: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /tellraw: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /time: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /timings: Records timings for all plugin events
[15:34:13] [Server thread/INFO]: /title: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /tm: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /tp: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /trigger: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /version: Gets the version of this server including any plugins in use
[15:34:13] [Server thread/INFO]: /w: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /weather: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /whitelist: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /worldborder: A Mojang provided command.
[15:34:13] [Server thread/INFO]: /xp: A Mojang provided command.
>
```

Pulling and Registries and Caches

Caches

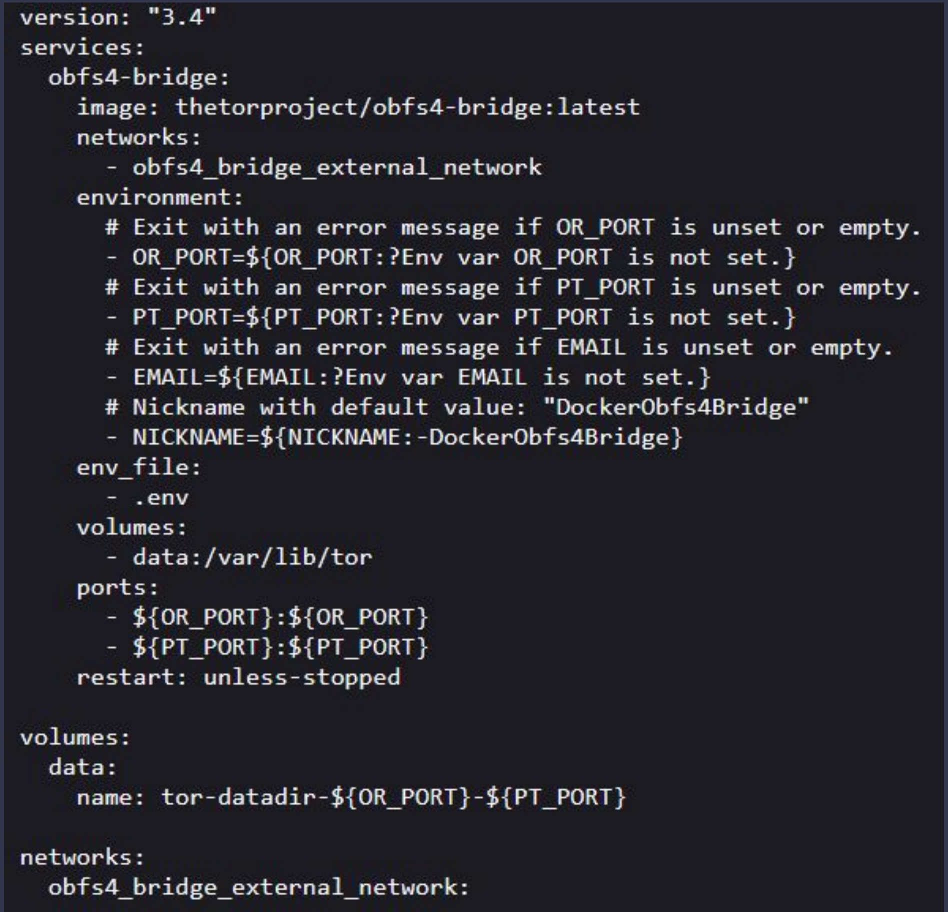
- You might not want to host a private hub (public image but slow speeds)
- Cache keeps a local copy of requested images
  - Image requested from cache
  - If not kept requested from hub and stored
  - Image sent to user



Docker compose

- YAML file
- Allows for creation of many resources at once: containers, volumes, networks etc.
- Compared to long command, easier to:
    - Create
    - Understand
    - Execute

sudo docker-compose up -d [NAME]



```
version: "3.4"
services:
  obfs4-bridge:
    image: thetorproject/obfs4-bridge:latest
    networks:
      - obfs4_bridge_external_network
    environment:
      # Exit with an error message if OR_PORT is unset or empty.
      - OR_PORT=${OR_PORT:?Env var OR_PORT is not set.}
      # Exit with an error message if PT_PORT is unset or empty.
      - PT_PORT=${PT_PORT:?Env var PT_PORT is not set.}
      # Exit with an error message if EMAIL is unset or empty.
      - EMAIL=${EMAIL:?Env var EMAIL is not set.}
      # Nickname with default value: "DockerObfs4Bridge"
      - NICKNAME=${NICKNAME:-DockerObfs4Bridge}
    env_file:
      - .env
    volumes:
      - data:/var/lib/tor
    ports:
      - ${OR_PORT}:${OR_PORT}
      - ${PT_PORT}:${PT_PORT}
    restart: unless-stopped

volumes:
  data:
    name: tor-datadir-${OR_PORT}-${PT_PORT}

networks:
  obfs4_bridge_external_network:
```

Storage

Container file system works like normal computer - this is lost when container is removed

For permanent storage:

- Bindings: Folder in container synced to folder in host system
  - Like a shared folder in a VM
  - Easy to access files in host system (manually or other software)
- Volumes: Storage area/drive only for docker
  - Bit like a USB drive
  - Can be shared between containers
  - Slightly better performance

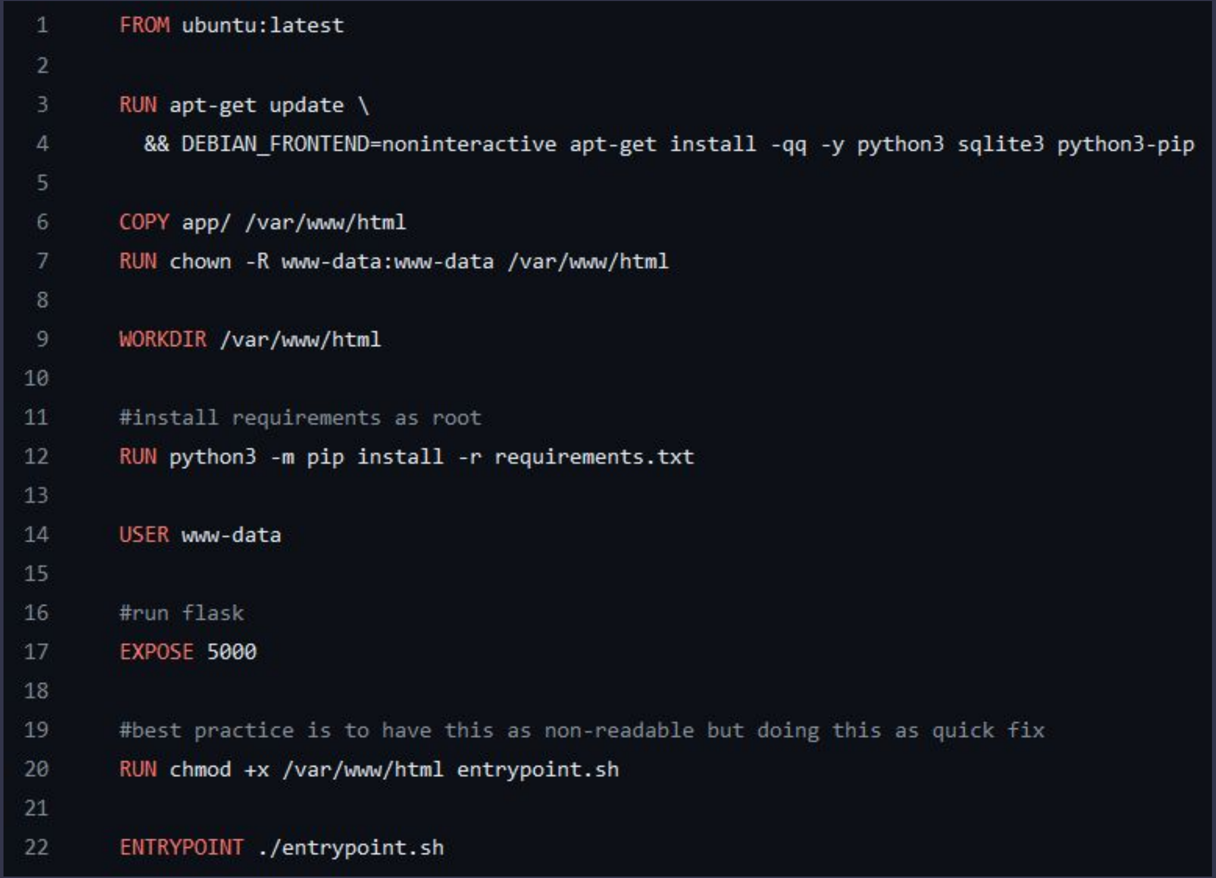
Networking

- Docker containers are put on their own little subnet
- Can be networked together
- Most common is forwarding ports from the host
    - -p host:container

Can connect to other peoples docker containers

- Multiple servers can be hosted on one device
- Game servers, web servers, etc

Dockerfile



```
FROM ubuntu:latest

RUN apt-get update \
  && DEBIAN_FRONTEND=noninteractive apt-get install -qq -y python3 sqlite3 python3-pip

COPY app/ /var/www/html
RUN chown -R www-data:www-data /var/www/html

WORKDIR /var/www/html

#install requirements as root
RUN python3 -m pip install -r requirements.txt

USER www-data

#run flask
EXPOSE 5000

#best practice is to have this as non-readable but doing this as quick fix
RUN chmod +x /var/www/html entrypoint.sh

ENTRYPOINT ./entrypoint.sh
```

The Dockerfile is the centerpiece of Docker, it is one of the scripts that tells the Docker daemon how to construct our containers.

Docker uses a combination of its own syntax and also bash to setup the environment.

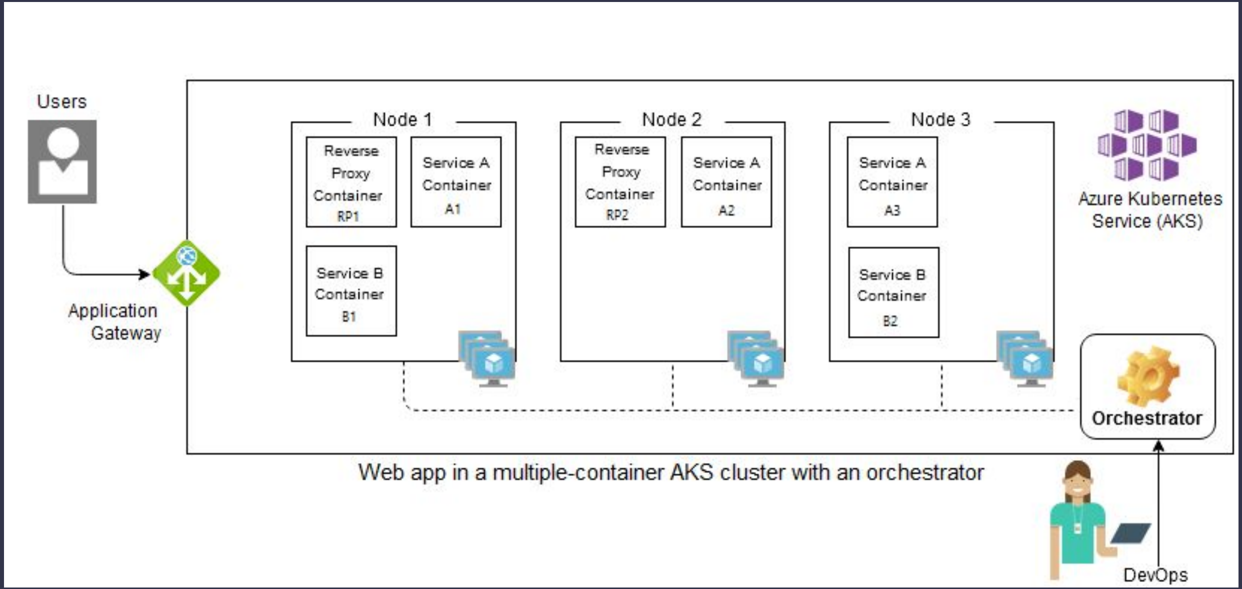
- FROM - Specify the base image to use, e.g. ubuntu or flask
- COPY - Copy files from the host to the container
- RUN - runs a command as a user, by default it runs as root in bash
- WORKDIR - Specify the working directory
- USER - Specify the user to change to
- ENV - Set an environment variable
- EXPOSE - Allow a port to be connected to
- ENTRYPOINT - The script to run once the container has started

Orchestration

- Scaling
- Load balancing
- Failures/portability
- Secure communication
- Easy deployment including across environments

Kubernetes is an open source container orchestration tool.

Works on containers being microservices where they do a very specialised role.



Web app in a multiple-container AKS cluster with an orchestrator

Security

Update and give minimum permissions.

Misconfiguration:

- Ports/Networking - access to network potentially including the host, other containers and other devices
- Volumes - shared storage with other containers
- Bindings - shared storage with host

Breaking the container - (next slide)

You may also be able to find dockerfile or docker-compose.yml files that leak credentials

Docker uses the hosts kernel so if there is a vulnerability in that then the host will be accessible

Vulnerabilities

You can find best practices here: https://cheatsheetseries.owasp.org/cheatsheets/Docker_Security_Cheat_Sheet.html

If you are a member of the docker group, then you can easily gain root by exploiting the volumes ability to mount the host operating system to a container.

If the docker socket (`find / -name docker.sock 2>/dev/null`) is mounted inside the container you can use it to escape.

Running a docker container with `--privileged` can allow the container to interact with host ports, capabilities and overall lead to code execution on the host as root.

And more…

Articles: CVE-2019-5736 (https://unit42.paloaltonetworks.com/breaking-docker-via-runc-explaining-cve-2019-5736/), Privileged Flag Exploit Example (https://0xdf.gitlab.io/2021/05/15/htb-ready.html#shell-as-root-host), CVE-2019-5736 Example (https://0xdf.gitlab.io/2021/07/31/htb-thenotebook.html#shell-as-root), https://book.hacktricks.xyz/linux-unix/privilege-escalation/docker-breakout/docker-breakout-privilege-escalation

Connecting to other people's containers

If you are testing a networked service like a game server, you should be able to connect to other people's machines.

To get your IP address you can run 'ipconfig' (Windows) / 'ifconfig en0' (mac) / 'ifconfig wlan0' (linux)

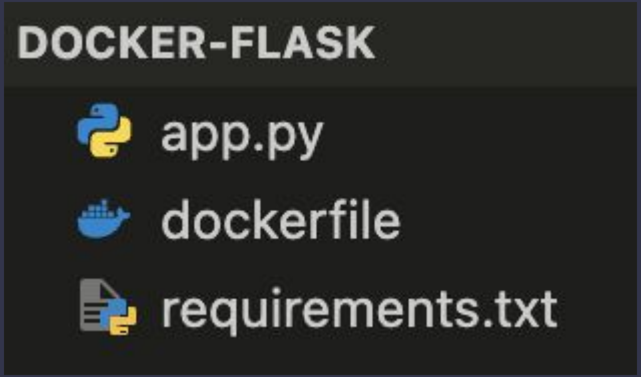
Other people can now connect to your linux container by going to YOUR_IP:PORT, where the port is the port of the docker container you exposed

Practical

Create a new folder (Docker-Flask)

Create three files

- app.py
- dockerfile - must be lowercase
- requirements.txt


DOCKER-FLASK
app.py
dockerfile
requirements.txt

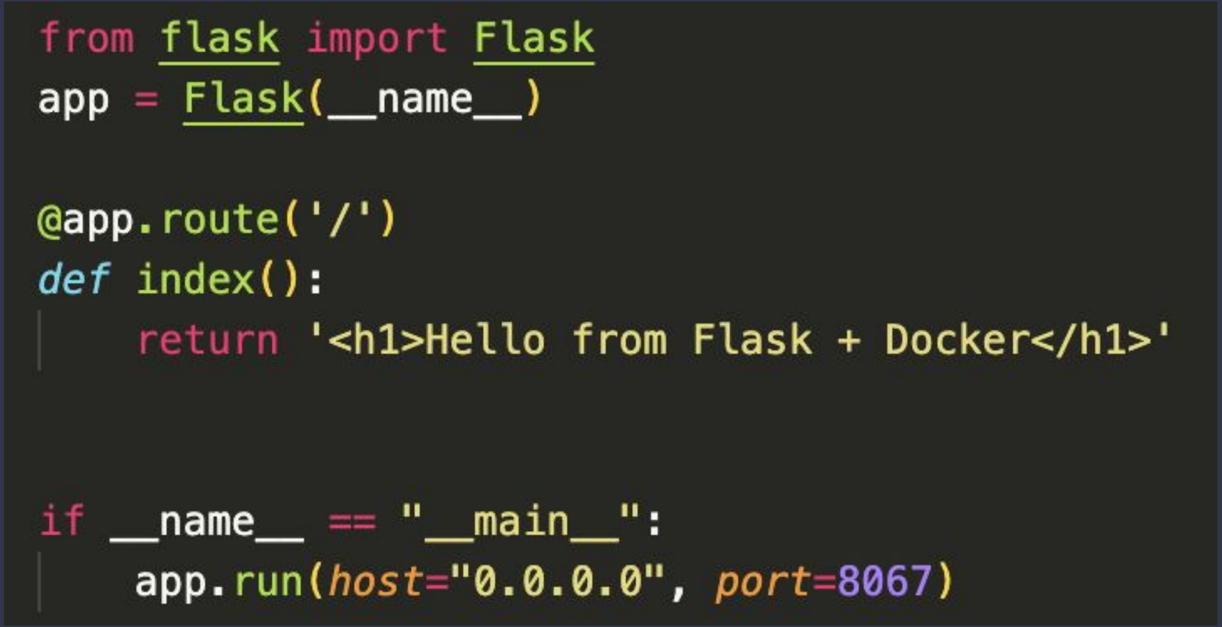
Practical

Requirements.txt

- Add the word ‘flask’ and save the file

App.py

- Customise the message
- Remember the value you set port to

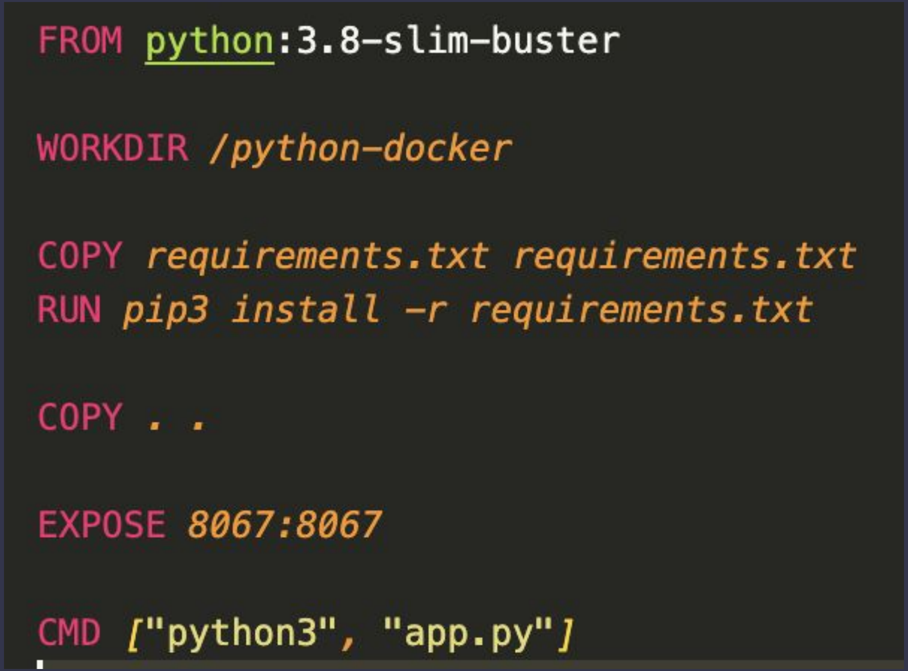


```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
    return '<h1>Hello from Flask + Docker</h1>'


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8067)
```

Practical

dockerfile

- Using slim version of python to make containerising quicker
- Not quite done yet

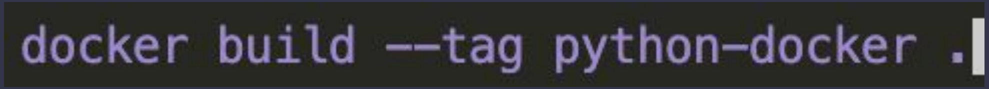


```
FROM python:3.8-slim-buster

WORKDIR /python-docker

COPY requirements.txt requirements.txt
RUN pip3 install -r requirements.txt

COPY . .

EXPOSE 8067:8067

CMD ["python3", "app.py"]
```

Practical

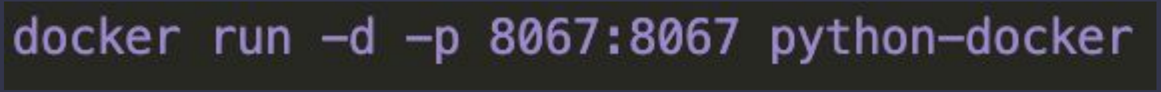
Terminal

- Move into the directory of your project
  - cd
- Run this command to build the docker container
  - Creates the image from the dockerfile with the ‘tag’ python-docker



```
docker build --tag python-docker .
```

- Run this command to
  - -d -> run in detached mode so we can still use the terminal
  - -p -> maps port 8067 on our device (use whatever value you set previously here) to an external port (doesn’t have to be the same)
  - ‘Python-docker’ <- tag for the docker image



```
docker run -d -p 8067:8067 python-docker
```

More Practical

- Run a more interesting container:
  - itzg/minecraft-server
  - wordpress
  - httpd
  - Find one that interests you…
- Create your own Dockerfile
  - Using your own web and internet project upload and connect to each others

Ask us for any help or questions you may have.

Be careful what you host. Ask peoples permission before connecting to things.

https://docs.docker.com/get-started/

Upcoming Sessions

What's up next?
www.shefesh.com/sessions
shefcompsoc.uk/events/

CompSoc Upcoming Events:
AGM - Wednesday 2nd April

Kubernetes Workshop - Wednesday 30th April

Summer Ball - Friday 16th May

Any Questions?

www.shefesh.com

Thanks for coming!