



# Cyber Security



# Workshop Docker

- 01 Was ist Docker?**
- 02 Vorteile von Docker**
- 03 Unterschiede zwischen Docker und VMs**
- 04 Voraussetzungen und Installation**
- 05 Praxis**



# 01

# Was ist Docker?



# Was ist Docker?

- Plattform zur Containerisierung (siehe unten) von Anwendungen
  - Leichtgewichtige Alternative zu virtuellen Maschinen
  - Open Source, weit verbreitet in DevOps und IT-Sicherheit
  - Container = isolierte Prozesse mit eigenem Dateisystem
- 
- Erklärung: Containerisierung  
Bei der Containerisierung werden Anwendungen in abgetrennten Bereichen auf einem bereits laufenden Betriebssystem ausgeführt.  
Diese speziellen, getrennten Bereiche werden als Container bezeichnet und bringen bestimmte Vorteile mit sich. Siehe nächster Slide.



# 02

## Vorteile von Docker



# Vorteile von Docker

- Jede Anwendung läuft in ihrem eigenen Container (unabhängig von anderen Prozessen) und bietet somit mehr Sicherheit
- Plattformunabhängigkeit:  
Kann überall ausgeführt werden, solange eine Container-Engine (z. B. Docker) installiert ist.
- Schnelle Bereitstellung von Anwendungen
- Ressourcenfreundlich und skalierbar

Daher ist Docker Ideal für Tests, Schulungen und Analyseumgebungen.



# 03

## Unterschiede zwischen Docker und VMs





# Docker vs. virtuelle Maschinen

Eigenschaft	Docker-Container	Virtuelle Maschinen
Startzeit	Sekunden	Minuten
Overhead	Gering	Hoch
Isolierung	Prozessbasiert	Meist Hardwareemulation
Portabilität	Hoch	Mittel



# 04

# Voraussetzungen und Installation



# Voraussetzungen

- Läuft auf allen gängigen Betriebssystemen
- Im BIOS das aktivierte Feature:  
VT-x bzw. AMD-V
- Grundlegendes Verständnis für die CLI
- Netzwerk Kenntnisse
- Weitere, betriebssystemspezifische Voraussetzungen finden sich in der Dokumentation:  
<https://docs.docker.com/engine/install/>



# 05

# Praxis



# Praxis: Installation

(auf Kali bzw. Debian Linux)

## Sicherheitshinweis:

- If you use ufw or firewalld to manage firewall settings, be aware that when you expose container ports using Docker, these ports bypass your firewall rules. For more information, refer to Docker and ufw.
- Docker is only compatible with iptables-nft and iptables-legacy. Firewall rules created with nft are not supported on a system with Docker installed.

Quelle:

[<https://docs.docker.com/engine/install/debian/>]



# Praxis: Installation

(auf Kali bzw. Debian Linux)

- Schritt 1:  
Alle Pakete, welche Konflikte verursachen könnten, müssen deinstalliert werden. Dies wird mit folgender Kommandozeile erreicht:

```
for pkg in docker.io docker-doc docker-compose podman-docker  
containerd runc; do sudo apt-get remove $pkg; done
```



# Praxis: Installation

(auf Kali bzw. Debian Linux)

- Schritt 2:  
Installation der Docker Engine. Dazu folgen Sie den Schritten auf folgender Seite:

<https://docs.docker.com/engine/install/debian/>



# Praxis: Installation

(auf Kali bzw. Debian Linux)

- Schritt 2: Wichtiger Hinweis: (speziell für Kali und andere Drivate)  
Im ersten Schritt der Installation, muss in der Zeile

`$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \`

`$VERSION_CODENAME` durch den realen Namen ersetzt werden.

```
(kali㉿kali)-[~]  
$ echo \  
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker  
r.com/linux/debian \  
$(. /etc/os-release && echo "bookworm") stable"__|\  
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  
sudo apt-get update
```





# Praxis: Test der Installation

(auf Kali bzw. Debian Linux)

- Schritt 3: Überprüfen der Installation bzw. Ausgabe der Version.

Kommando:

```
(kali@kali)-[~]  
$ docker --version  
Docker version 28.2.2, build e6534b4
```



# Praxis: Test der Installation

(auf Kali bzw. Debian Linux)

- Schritt 3: Überprüfen der Installation.

... oder mit dem klassischen Hello World:

```
sudo docker run hello-world
```

kann überprüft werden ob Docker korrekt installiert wurde.

Beispielhafte Ausgabe nach erfolgter Installation:

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



# Praxis:

# Docker Kommandos

- Docker Kommandos sind in verschiedene Bereiche aufgeteilt:
  - Common Commands
  - Management Commands
  - Swarm Commands
  - Commands
- Des Weiteren existieren diverse Optionen für
  - den Debug Modus
  - den Log Level
  - ...



# Praxis:

# Docker Kommandos

- Wenn bei der Ausführung der Docker Kommandos eine „permission denied“ Fehlermeldung erscheint kann das Problem mit

`sudo usermod -aG docker $USER` (z.B. „kali“)

behoben werden.



# Praxis: einige wichtige Docker Kommandos

- `docker ps -a`      # alle Container anzeigen
- `docker start ID`    # Container starten
- `docker stop ID`     # Container stoppen
- `docker rm ID`        # Container löschen



# Praxis: weitere wichtige Docker Kommandos

- Damit an dieser Stelle nicht alle Befehle aufgelistet werden müssen, erfolgt der Hinweis auf das

**CLI Cheat Sheet:**

**[https://docs.docker.com/get-started/docker\\_cheatsheet.pdf](https://docs.docker.com/get-started/docker_cheatsheet.pdf)**



# Praxis:

## Nutzung vorhandener Container

- Vorgefertigte Images können von der Webseite

`hub.docker.com`

dem sogenannten Docker Hub heruntergeladen werden.

- Aus Sicherheitsgründen auf die Quelle achten!  
Zum Beispiel indem die Option „official Docker Images“ ausgewählt wird.



# Praxis:

## Nutzung vorhandener Container

- Images von Docker Hub laden: (mittels ‚pull‘ und ‚run‘)

```
docker pull kalilinux/kali-rollingkalilinux/kali-rolling  
/bin/bash  
docker run -it
```





# Praxis:

## Nutzung von „compose“

- Docker compose vereinfacht den Umgang mit mehreren Containern:

Es ermöglicht multi-container Umgebungen zu definieren, laufen zu lassen und zu verwalten.

```
docker compose up -d
```

die Option steht für „detatch“ und sorgt dafür, daß die Linux Console weiter benutzt werden kann und nicht für die Dauer der Sitzung blockiert ist. Container laufen im Hintergrund.

- Die Datei „docker-compose.yml“ enthält die notwendigen Konfigurationsdaten



# Praxis: compose YAML- Datei

## (erster Teil)

```
# Die Docker-Compose-Zusammenstellung für Pi-hole (https://hub.docker.com/r/pihole/pihole)
services:
  pihole:
    container_name: pihole
    image: pihole/pihole:latest
    ports:
      - "53:53/tcp,,
      - "53:53/udp,,
      - "67:67/udp,,
      - "80:80/tcp,,
    environment:
      TZ: 'Europe/Berlin'
      WEBPASSWORD: ',*****'          <- Hier ein sicheres Passwort eintragen
    volumes:
      - './etc-pihole:/etc/pihole'
      - './etc-dnsmasq.d:/etc/dnsmasq.d'
    cap_add:
      - NET_ADMIN
    restart: unless-stopped
```



# Praxis: compose YAML- Datei

## (zweiter Teil)

```
portainer:  
  image: portainer/portainer-ce  
  ports:  
    - 9000:9000  
  volumes:  
    - /var/run/docker.sock:/var/run/docker.sock  
    - ./portainer_data:/data    restart: always
```



# Praxis:

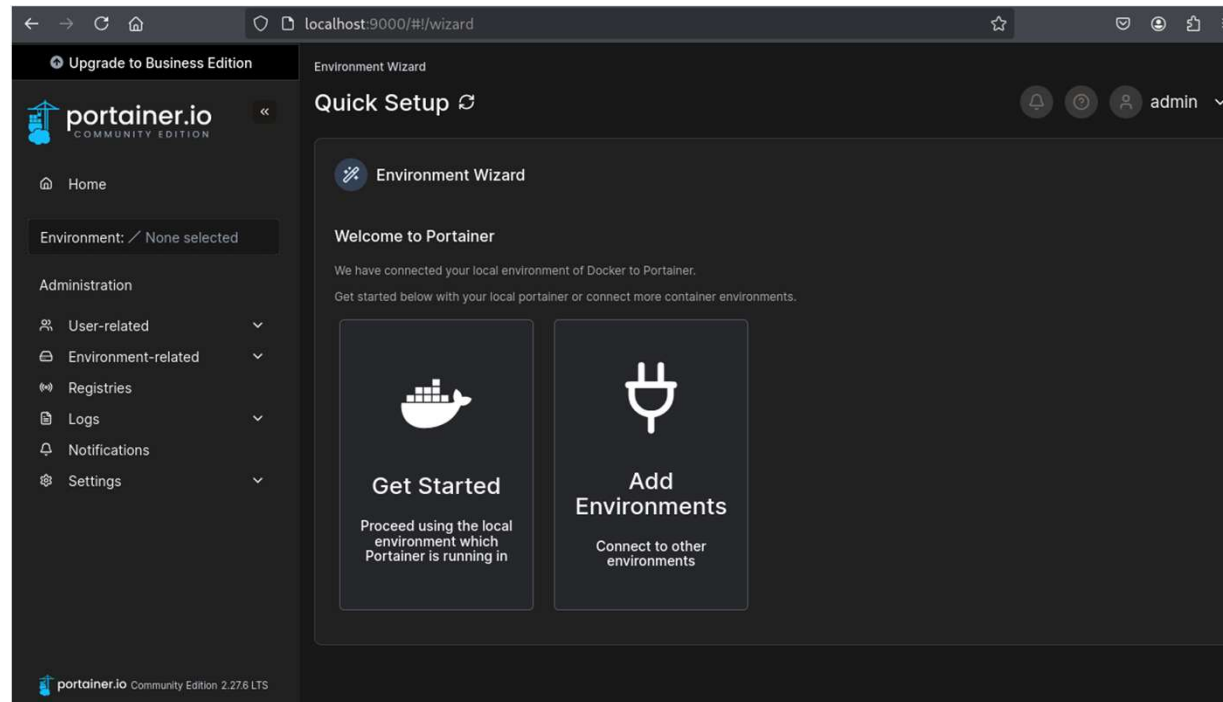
- Nachdem die vorhin gezeigte YAML Datei verarbeitet wurde (docker compose up -d), können Container auch über eine Weboberfläche administriert werden. Dies erfolgt durch den Container „portainer“.
- Aufruf der Weboberfläche erfolgt in einem Browser auf den Localhost, Port 9000 (im Standard):

<http://localhost:9000>



# Praxis:

- Screenshot des UI von „Portainer“



# Praxis:

## Updates einspielen bzw. holen

- Befehl, um Container zu updaten:

```
docker compose pull
```

- Oder man installiert „watchtower“, welches automatisch nach Updates sucht



DANKE!

# Gibt es noch Fragen?





# CloudCommand