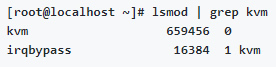
Static IP + SSH Server

* Statische IP-Adresse einrichten uter **/etc/network/interfaces**
* Nano installieren mit **apk add nano** und alle verfügbaren Packages aktivieren unter **nano /etc/apk/repositories** und auskommentieren, um später wichtige Programme für KVM und Docker installieren zu können
* SSH Server eingerichtet mit Port 22 und root-Rechte akzeptieren in Konfigurationsdatei **/etc/sshd/sshd\_config**

KVM

* Nested Virtualisierung in Virtualbox aktivieren: In **Windows**, go to VirtualBox installation folders -> type cmd on the bar (it will pop up cmd in that folder) -> type **VBoxManage modifyvm YourVirtualBoxName --nested-hw-virt on**
* Prüfen, ob KVM funktionieren müsste: 



* apk add libvirt-daemon qemu-img qemu-system-x86\_64 qemu-modules -> Hinzufügen von libvirt und qemu auf ein x86-64 System
* apk add dbus polkit virt-manager terminus-font -> Virtmanager hinzufügen

Docker

* Docker Installation mithilfe **apk add docker**
* **rc-update add docker boot -> Docker Deamon beim Booten starten**
* **service docker start** -> Docker Deamon im aktuellen Zeitpunkt starten
* **docker run hello-world** -> Beispiel Docker zum Testen, ob Docker funktioniert
* **Weitere Schritte zur Installation von NGINX und Portainer**: Insgesamt 3 verschiedene Möglichkeiten, um ans Ziel zu kommen:

1. Alles über die Bash installieren über docker Befehle

2. Portainer zuerst installieren und dann über GUI NGINX Container mit Volume und Image erstellen

3. Docker Compose File erstellen und dann zusammen beide Container über dieses Script direkt und schnell installieren

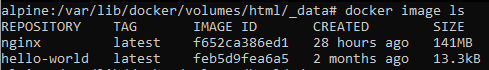
**MÖGLICHKEIT 1: EINGEBEN VON BEFEHLEN IN DER BASH**

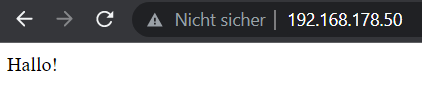
INSTALLATION VON NGINX

* **docker run –name mein-webserver -d -p 80:80 nginx** -> Startet den Container nginx mit dem Namen „mein Webserver“, läuft im Hintergrund (deamonize -d) und wird aus Port 80 konfiguriert (Port 80 wird auf 80 weitergeleitet)
* **docker ps** -> Zeigt alle aktuell laufenden Container an



* **docker image ls** -> Zeigt alle installierten Container an



* **docker stop mein-webserver**  -> Stoppt den laufenden Container wieder
* Der Container nginx wird nun auch in der Prozessliste aufgeführt: **ps axu | grep nginx**
* **docker start mein-webserver** -> Startet den Container
* **docker exec** **-it mein-webserver bash** -> Startet das Programm bash innerhalb des Containers mit dem Namen mein-webserver (nginx) und -it um eine interatkive Session auszuführen. Nun befindet man sich als Root im Verzeichnis des nginx Containers.
* Mehrere Informationen zum verwendeten Image (hier nginx) findet man auf der Webseite hub.docker.com
* Um nun Dateien in den Docker abzulegen (Webseite in den Docker abspeichern) muss man diesen Inhalt für die Webseite in den Pfad **/usr/share/nginx/html** abspeichern und darin kann dann z.B. die index.html Datei bearbeitet werden
* Es gibt jedoch keinen Standardeditor innerhalb des Containers, da es sich um ein Minimalsystem handelt! (u.a. auch Sicherheitsgründe). Ich habe stattdessen **echo „hallo!“ > index.html** probiert und dann im Browser die IP-Adresse des Alpine Virtualmaschine eingeben. Der Webserver funktioniert soweit.
* Normalerweise verwendet man nicht das Programm bash, um auf die jeweiligen Container zuzugreifen, sondern Volumes (Verzeichnisse auf dem Hostrechner, die vom Docker als sogenannte Overlays in den Container hineingemountet werden). Zudem wird die Änderung nicht gespeichert wenn der Docker z.B. neu gestartet wird. Dies geht nur mithilfe Volumes!
* **docker volume create**  -> Erzeugt ein Volume für den Container mit generiertem langen Zahlencode
* **docker volume create html** -> Erzeugt ein Volume mit dem Namen html (Besser lesbar als Befehl zuvor)
* **docker volume ls**  -> Zeigt alle Volumes für den Container an
* **docker run --name webserver -p 80:80 -v html:/usr/share/nginx/html:ro -d nginx** -> Erstellt nun einen Container wie zuvor mit dem Namen Webserver und dem dazugehörigen Volume html (zuvor erstellen). :ro sagt aus, dass Ngnix nicht auf die statischen HTML-Daten schreiben soll.
* Über das Hostsystem können nun die Daten für den Container verändert werden über den Pfad **/var/lib/docker/volumes** und hier kann man sehen, dass sich darin das Volume html befindet. Innerhalb des Ordners befindet sich \_data und darin befindet sich die index.html Datei, welche nun auf dem Hostsystem mithilfe nano bearbeitet werden kann. Diese Daten bleiben dann erhalten, auch wenn man den Container stoppt und wieder startet. Auch wenn man den Container löscht und einen neuen erstellt mit dem gleichen Volume html, bleibt die Webseite dadurch gleich! 🡪 Container neu aber gleicher Inhalt
* Möchte man seine Volume in einen anderen Pfad ablegen, so gibt man statt html den Pfad an:

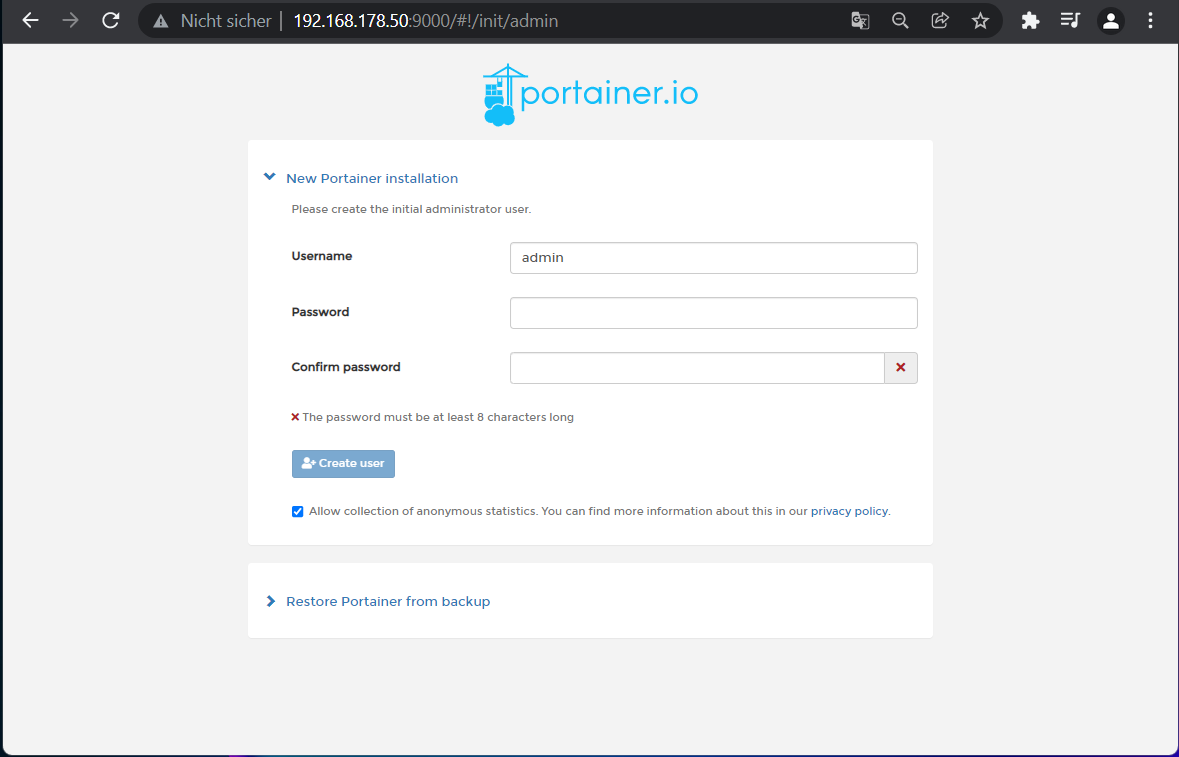
**docker volume create /home/tobias/html** Jedoch darauf achten, dass keine Rechte benötigt werden, damit docker auf den Pfad auch zugreifen kann!

* **docker volume rm html** -> Löscht das Volume wieder
* **docker volume prune** -> Löscht alle Volumes, die mit keinem Container mehr verknüpft sind (Falls z.B. Container gelöscht wurde)

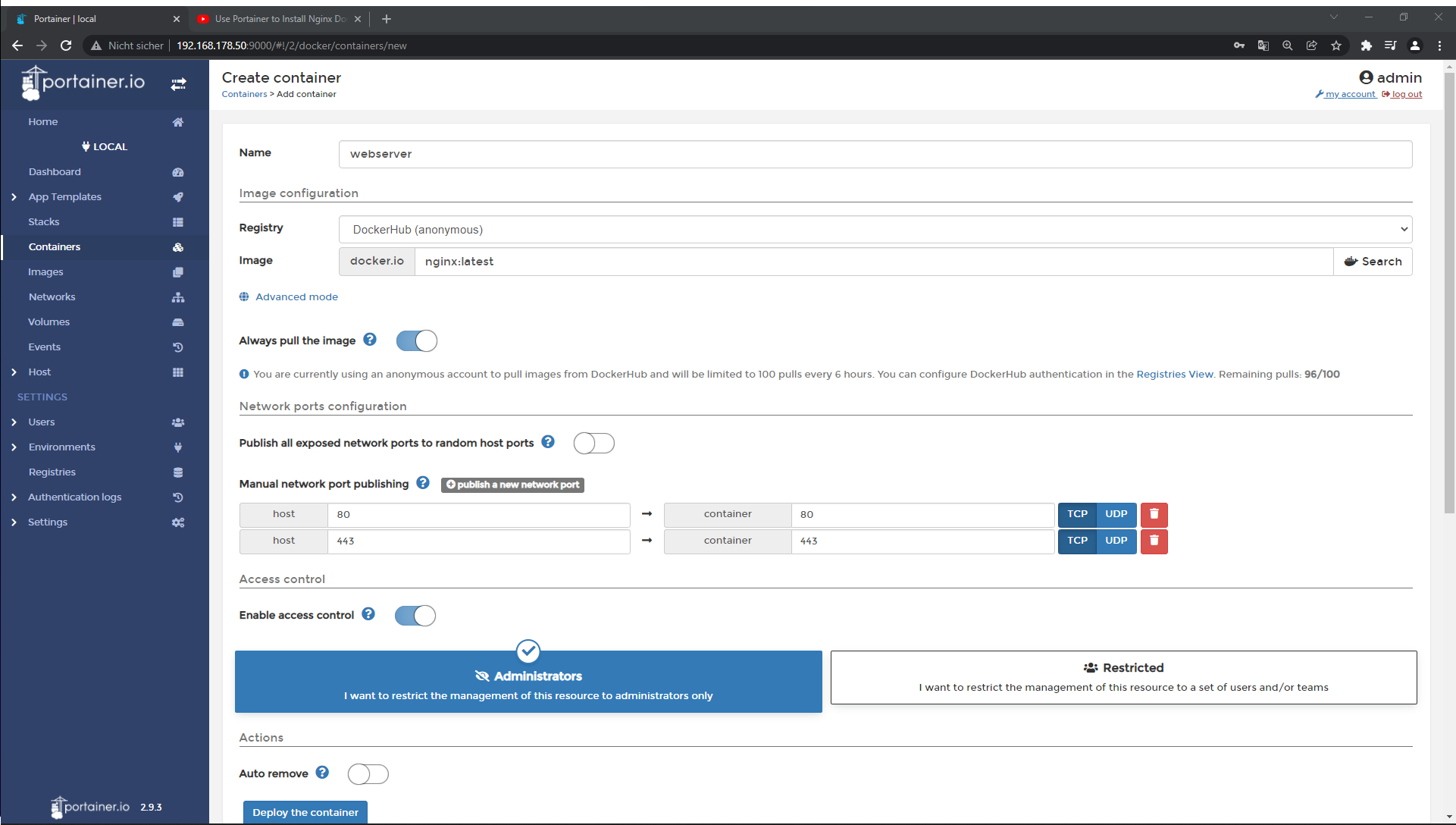
Container vernetzen

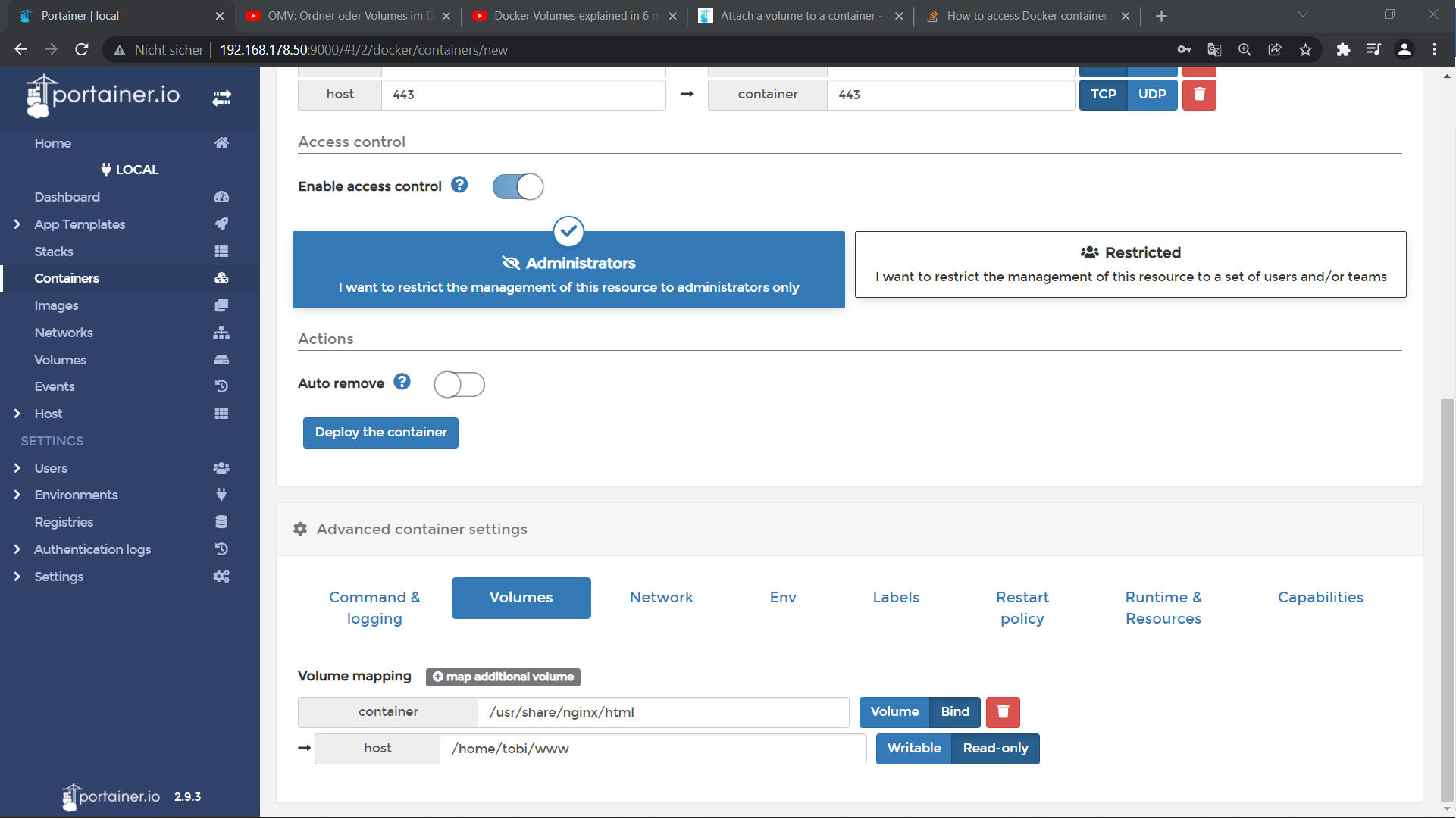
* Der Docker Deamon erstellt beim Start ein eigenes Netzwerkgerät **docker0**, einsehbar mit **ifconfig docker0**
* **Docker network ls** -> Listet die verschiedenen Netzwerktypen auf (Bridge, Host, none)
* **docker network create --driver bridge webserver-bridge** 🡪 Erstellt ein neues Netzwerk mit Typ Brücke und dem Namen webserver-bridge
* Dieses Netzwerk kann dann bei der Erstellung eines neuen Containers mitdrangehängt werden: **docker run --name webserver --network=webserver-bridge -p 80:80 -v html:/usr/share/nginx/html:ro -d nginx**
* **Docker network inspect webserver-bridge**  -> Zeigt genauere Informationen zum Netzwerk

INSTALLATION VON PORTAINER

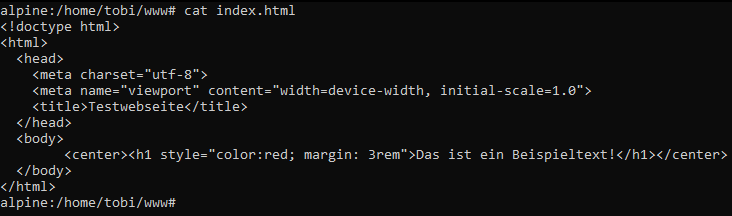
* docker volume create portainer\_data -> Volumen erzeugen, wo der Portainer Serer seine Daten abspeichern kann. Erhält dann einige Logdateien, Konfigurationsdateien.
* **docker run -d -p 8000:8000 -p 9000:9000 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer\_data:/data cr.portainer.io/portainer/portainer-ce** 🡪 Installiert den Portainer, mehr Infos zu den Optionen unter: <https://kayomo.de/blog/portainer-unter-docker-installieren/>
* 

**MÖGLICHKEIT 2: Zuerst Portainer einrichten und dann über GUI NGINX einrichten**





Unter /home/tobi/www wird dann die index.html erstellt:



Docker Compose

* Übersichtlicher als die typischen run-Befehle
* **apk add docker-compose** -> Installiert docker-compose
* **docker-compose.yml** Datei im lokalen Arbeitsverzeichnis erstellen und dann **docker-compose up** ausführen, um die die Container zu erstellen, die innerhalb der Datei angegeben wurden. Es müssen immer 2 Zeilen Abstand sein, falls eine Einrückung erfolgt!