A close-up of water droplets

Description automatically generated with low confidence**Betriebsysteme Projekt WS22/23**

**Thema 2:**

**Clustering mit Docker**

**Dokumentation**

**Von Ahmed Abdaal und Jenny Rötzer**

**Inhaltsverzeichnis**

***1. VM einrichten***

***2. Installation von Docker***

***3. NFS***

***4. Portainer / Visualizer***

***5. Docker Swarm einrichten***

***6. ???***

***7. ???***

***8. Test***

***9. Quellen***

***Alle lila-gefärbten Themen kannst du dann ausformulieren und erklären.***

***Du kannst entscheiden ob du Portainer benutzen willst oder Visualizer. Du kannst auch selber entscheiden auf welchen Platz der Stichpunkt kommt, also ob es vor NFS kommt oder danach.***

***Bei den Fragezeichen schauen wir uns erstmal die Lösung von den Dude an, damit wir was checken und dann schreiben wir was rein.***

**1. VM einrichten**

**2. Installation von Docker:**

Zunächst laden wir Docker auf den Virtuellen Maschinen runter. Der Prozess ist auf allen vier Maschinen ist derselbe.

Damit wir Docker installieren können müssen wir ein Verzeichnis für Docker auf unseren Maschinen einrichten.

**Schritt 1:** Pakete aktualisieren und dann neue installieren, damit *apt* das Verzeichnis

über HTTPS benutzen kann

$ *sudo apt-get update*

$ *sudo apt-get install \*

*ca-certificates \*

*curl \*

*gnupg \*

*lsb-release*

**Schritt 2:** Docker GPG key hinzufügen

$ *sudo mkdir -p /etc/apt/keyrings*

$ *curl -fsSL https://download.docker.com/linux/debian/gpg | sudo*

*gpg --dearmor -o /etc/apt/keyrings/docker.gpg*

**Schritt 3:** Verzeichnis anlegen

$ *echo \*

*"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]*

*https://download.docker.com/linux/debian \*

*$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >*

*/dev/null*

Nachdem wir das Verzeichnis eingerichtet haben, installieren wir jetzt die Docker Engine.

**Schritt 1:** apt Pakete aktualisieren

$ *sudo apt-get update*

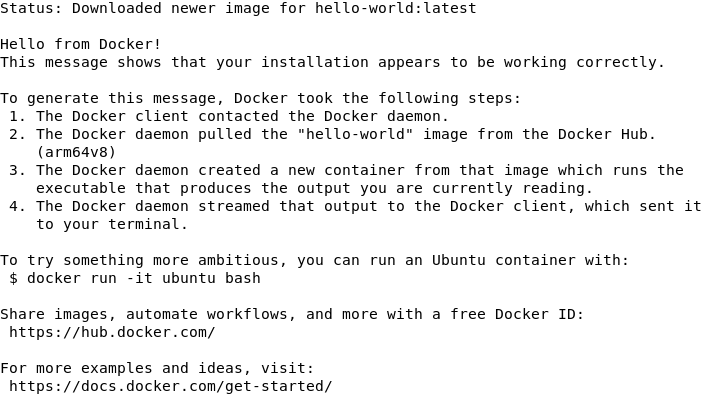
**Schritt 2:** die neuste Version der Docker Engine herunterladen

$ *sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-*

*plugin*

**Schritt 3:** testen mit*hello-world* image

$ *sudo docker run hello-world*



Bei erfolgreichem Download sollte die Ausgabe so ausschauen:

**Docker als non root-user benutzen:**

Vor jedem Docker Befehl müssen wir *sudo* anhangen, weil der Docker Deamon immer als *root* User läuft. Um das zu vermeiden, tun wir unseren User in der Docker Gruppe rein.

**Schritt 1:** Gruppe erstellen (falls sie noch nicht existiert)

$ *sudo groupadd docker*

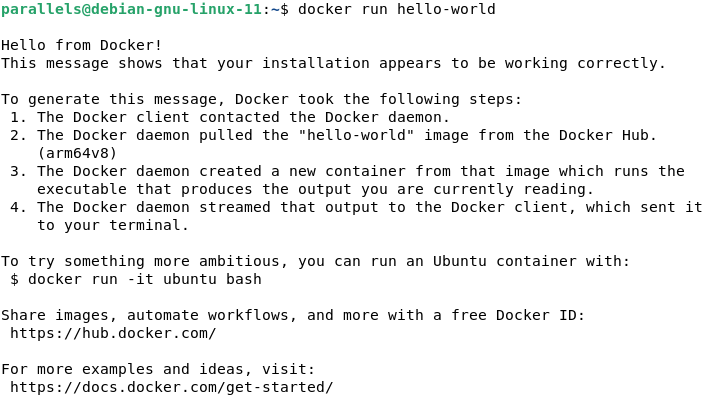
**Schritt 2:** User in der Docker Gruppe hinzufügen

$ *sudo usermod -aG docker $USER*

Danach sollte man seine VM neu starten, damit die Veränderungen gültig werden.

**Schritt 3:** *hello-world* image starten ohne sudo

$ *docker run hello-world*



Jetzt können wir ohne sudo docker benutzen, was uns später vieles erleichtert.

**3. NFS**

Wir benutzen *Network File System* als unser gemeinsamen Speichermittel für unseren Swarm Cluster. Dieses Protocol benutzt das Server/Client Prinzip. Eine Maschine ist der Server (Manager) und die anderen sind die Clients (Worker). Dabei sind, wie man schon ahnt, alle Daten auf dem Server gespeichert und die Clients können darauf zugreifen, so als wären die Dateien lokal gespeichert.

Dabei müsst ihr darauf achten, dass die Einrichtung von NFS auf der Server-Maschine anders ist als bei den Client-Maschinen.

**Schritt 1:** Packete installieren

auf der Server-Maschine

$ *sudo apt update*

$ *sudo apt-get install nfs-kernel-server*

auf den Client-Maschinen

$ *sudo apt update*

$ *sudo apt-get install nfs-common*

**Schritt 2:** Verzeichnis erstellen und Rechte ändern

auf der Server-Maschine

$ *mkdir nfs-share*

$ *sudo chown nobody:nogroup nfs-share*

auf den Client-Maschinen

$ *mkdir nfs-server*

**Schritt 3:** NFS freigeben

auf der Server-Maschine

$ *sudo nano /etc/exports*

Graphical user interface, text

Description automatically generated

Nachdem öffnen von */etc/exports* sollte euer Bildschirm so ausschauen. Die drinstehenden Kommentare zeigen uns wie die Eingabe erfolgen soll.

Wir ergänzen ganz unten:

*/home/parallels/nfs-share 10.211.55.13(rw,sync,no\_root\_squash,no\_subtree\_check)*

Shareoptionen

Client\_ip

Pfad zum Verzeichnis auf Host

**Wichtig: kein Leerzeichen nach dem Komma machen!!!**

****

Der Pfad, der am Anfang steht */home/parallesl/nfs-share ist der Pfad zu dem geteilten Verzeichnis. Danach kommt die IP-Addresse oder Hostname des Clients, im unserem Fall 10.211.55.13. Die im klammer stehende Syntaxen sind die Shareoptionen.*

**rw:** *client kann Lesen und Schreiben*

**sync:** *Synchroner Datentransfer*

**no\_root\_squash:** *begrenzt die Rechte für einen SU per Remotezugriff*

**no\_subtree\_check:** *überprüft Unterverzeichnisbäumen nicht ab*

Wenn man mehrere Clients hat, so fügt man dasselbe wie oben aber mit der IP-Addresse des anderen Clients hinten an der Zeile an.

**Graphical user interface, text

Description automatically generated**

Nachdem wir die Datei gespeichert bzw. geschlossen haben, müssen wir den neuen Befehlen in der *exports*-Datei aktivieren.

auf der Server-Maschine

$ *sudo exportfs -ra*

Wir können uns die aktiven Maschinen auflisten

auf der Server-Maschine

$ *sudo exportfs -v*

Graphical user interface, text, application, email

Description automatically generatedAusgaben bei unserem Fall schaut so aus:

**(Schritt 4:** NFS Serverneustarten ) noch testen

auf der Server-Maschine

$ *sudo systemctl restart nfs-kernel-server*

**Schritt 5:** Firewall ändern

(mussen wir noch testen)

**Schritt 6:** Verzeichnisse *„mounten“*

auf den Client-Maschinen

Pfad zum Verzeichnis auf client

Pfad zum Verzeichnis auf Host

host\_ip

$ *sudo mount 10.211.55.15:/home/parallels/nfs-share /nfs-server*

Um es zu testen erstellen wir eine Datei mit *touch*:

auf den Client-Maschinen

$ *sudo touch nfs-server/testdatei*

Und wir lassen uns auf der Server-Maschine die Datei anzeigen:

Graphical user interface, text, application

Description automatically generatedauf der Server-Maschin

$ *ls -lh nfs-share/*

Normalerweise müss man den mount-Befehl jedes mal beim Neustarten der Maschine wieder eingeben, um das zu vermeiden schreiben wir was in der „fstab-Datei“ der Client-Maschinen rein.

auf den Client-Maschinen

$ *sudo nano /etc/fstab*

Graphical user interface

Description automatically generatedSo sollte bei euch der Editor ausschauen:

In der letzten Zeile schreiben wir:

10.211.55.15:/home/parallels/nfs-share /home/parallels/nfs-server nfs rsize=8192,wsize=8192 0 0

host\_ip

Dateisysteme checks

Systemsicherung

Optionen für die Lese- und Schreibegeschwindigkeit

Dateisystem

Pfad zum Verzeichnis auf Host

Pfad zum Verzeichnis auf Client

Graphical user interface

Description automatically generated

Jetzt müssen wir das Verzeichnis noch *„unmounten“*:

auf den Client-Maschinen

$ *sudo umount nfs-server*

Graphical user interface, text, application

Description automatically generatedDann starten wir unsere Maschine neu und sehen, dass das NFS beim Start schon funktioniert.

Diesen Prozess müssen wir auf jeder Client Maschine ausführen, damit alle ein gemeinsamen Speicher haben.

**4. Visualizer/ Portainer**

**5. Docker Swarm einrichten**

Einleitung

Vorausetzungen

* Docker installieren
  + Docker ohne root laufen
* Portainer installieren

Vorgehensweise

- Probleme

- Lösungen

- Herausforderungen

- test

Zusammenfassung

* In echten leben

Quellen

Deckbild: <https://rocketloop.de/wp-content/uploads/2021/05/clustering-machine-learning-comprehensive-guide.jpg>