

Docker und Containertechnologien

INHALT

1. EINLEITUNG
2. INSTALLATION und HÄUFIG VERWENDETE BEFEHLE
3. EXPOSING PORTS und PORT MAPPING
4. VOLUME MAPPING und VERKNÜPFUNG CONTAINERS
5. DOCKER NETZWERKTYPEN
6. ERSTELLEN UNSERES EIGENEN DOCKER DATEI UND IMAGES
7. VERÖFFENTLICHUNG DES NEUEN IMAGES IN DOCKER HUB
8. DOCKER COMPOSE
9. ZUSÄLTICHE INFORMATIONEN

1. EINLEITUNG

Ein Container ist eine Standardsoftwareeinheit, die Code und alle seine Abhängigkeiten verpackt, damit die Anwendung schnell und zuverlässig von einer Computerumgebung zur anderen läuft. Ein Docker-Container-Image ist ein einfaches, eigenständiges, ausführbares Softwarepaket, das alles enthält, was zum Ausführen einer Anwendung erforderlich ist: Code, Laufzeit, Systemtools, Systembibliotheken und Einstellungen.

Container-Images werden zur Laufzeit zu Containern, und im Fall von Docker-Containern werden Images zu Containern, wenn sie auf [Docker Engine](#) ausgeführt werden .

Warum Docker?

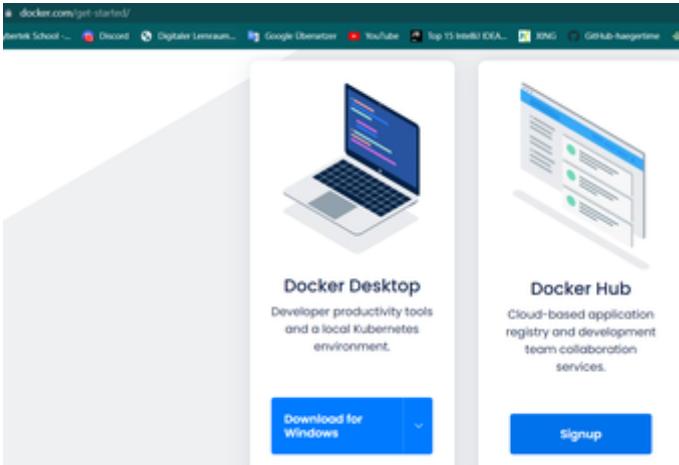
1. Prozessisolierung von der Anwendung ist möglich.
2. Die Anwendung kann eigenständig ausgeführt werden, ohne sie auf einem Computer oder Server zu installieren.
3. Um sicherzustellen, dass es kein Kommunikationsproblem zwischen den Entwicklern und DevOps gibt.
4. Um sicherzustellen, dass die Entwicklungs-, Test- und Produktionsumgebungen identisch sind.

2. INSTALLATION und HÄUFIG VERWENDETE BEFEHLE

Docker-Anforderungen und -Empfehlungen

Komponente	Richtlinien
Docker-Version	Docker Engine - Community-Version 18 oder höher ist erforderlich.
Betriebssysteme	MacOS, Linux, Windows 10 Professional oder Enterprise Edition. Docker verwendet einen Hypervisor mit einer VM, und der Hostserver muss Virtualisierung unterstützen. Da ältere Windows-Versionen und Windows 10 Home Edition Hyper-V nicht unterstützen, ist Windows 10 Professional oder Enterprise für Docker unter Windows erforderlich. Wenn Sie Docker CE unter Windows verwenden, konfigurieren Sie Docker außerdem für die Verwendung von Linux-Containern. Die Verwendung von Microsoft Windows-Containern wird nicht unterstützt, da sie Windows-API-Unterstützung für Windows-Containerdienstinstanzen bereitstellt.

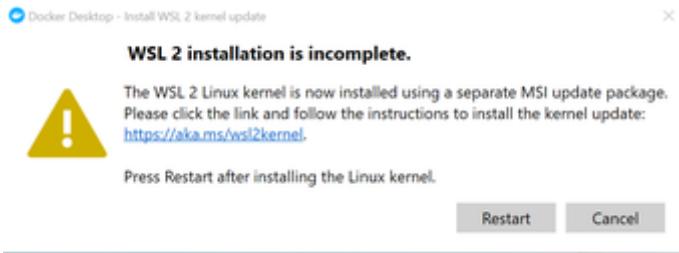
1. Installieren Sie Docker Desktop auf Ihrem Rechner (<https://www.docker.com/get-started/>). Um den Docker im CLI Modus zu starten, muss man Docker Desktop starten.



2. Schreiben Sie erste Befehl in CLI zum Aufruf des Anwendung Images (Beispiel Ubuntu) aus Docker Hub <https://hub.docker.com/>

```
# docker pull ubuntu
```

3. Wenn sich das Programm nicht öffnet, muss es von hier heruntergeladen werden. Wenn Sie diese Fehlermeldung haben, gehen Sie die Webseite unten und erledigen die Schritte, danach den Rechner neu starten.



[Manuelle Installationsschritte für ältere Versionen von WSL | Microsoft Docs](#)

4. Um die Liste der Images zu sehen;

```
# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mongo	latest	20106db9aa7a	4 days ago	700MB
ubuntu	latest	3f4714ee068a	4 days ago	77.8MB
docker/getting-started	latest	cb90f98fd791	2 weeks ago	28.8MB

```
C:\Users\Selami Demiral>
```

Um einen Image zu löschen ;

```
# docker rmi <IMAGE ID> oder # docker rmi -f <IMAGE ID>
```

Um mehrere Images zu löschen ;

```
# docker rmi -f $(docker images -a -q)
```

5. Um die Anwendung laufen zu lassen ;

```
# docker run -it ubuntu (Docker selbst gibt einen zufälligen Namen)
```

```

C:\Users\Selami Demiral>docker run -it ubuntu
root@26cccbda6473:/# ls
bin dev home lib32 libx32 mnt proc run srv tmp var
boot etc lib lib64 media opt root sbin sys usr
root@26cccbda6473:/# cd etc
root@26cccbda6473:/etc# ls
adduser.conf e2scrub.conf issue.net mtab rc1.d shells
alternatives environment kernel netconfig rc2.d skel
apt fstab ld.so.cache networks rc3.d subgid
bash.bashrc gai.conf ld.so.conf nsswitch.conf rc4.d subuid
bindresvport.blacklist group ld.so.conf.d opt rc5.d sysctl.conf
cron.d gshadow legal os-release rc6.d sysctl.d
cron.daily gss libaudit.conf pam.conf rcS.d systemd
debconf.conf host.conf login.defs pam.d resolv.conf terminfo
debian_version hostname logrotate.d passwd rmt update-motd.d
default hosts lsb-release profile security xattr.conf
deluser.conf init.d machine-id profile.d selinux
dpkg issue mke2fs.conf rc0.d shadow
root@26cccbda6473:/etc# cat os-release
PRETTY_NAME="Ubuntu 22.04 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04 (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
root@26cccbda6473:/etc#

```

6. Um laufende Containers zu sehen;

docker ps oder # docker container ls

Um die laufenden Container und Befehle für die Vergangenheit zu sehen ;

docker ps -a oder # docker container ls -a

Um die laufenden Container zu sehen ;

docker ps -q

```

C:\Users\Selami Demiral>docker run -it ubuntu
root@c60382eb9444:#

```

```

Eingabeaufforderung
Microsoft Windows [Version 10.0.22000.556]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Selami Demiral>docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
c60382eb9444 ubuntu "bash" About a minute ago Up About a minute epic_rubin
27078ad5c95e docker/getting-started "/docker-entrypoint...." 2 hours ago Up 2 hours 0.0.0.0:80->80/tcp tender_sinoussi

C:\Users\Selami Demiral>

```

Um ausführliche Informationen von laufende Anwendungen sehen,

docker inspect 13b....

```
C:\Users\Selami Demiral>docker inspeck 13b
docker: 'inspeck' is not a docker command.
See 'docker --help'

C:\Users\Selami Demiral>docker inspect 13b
[

{
    "Id": "13b9509ccc2fc5f0a2f77a58d7b3f247c4160062d02a7356ce3580637407aa7b",
    "Created": "2022-04-27T10:16:32.514Z",
    "Path": "docker-entrypoint.sh",
    "Args": [
        "mongod"
    ],
    "State": {
        "Status": "running",
        "Running": true,
        "Paused": false,
        "Restarting": false,
        "OOMKilled": false,
        "Dead": false,
        "Pid": 1503,
        "ExitCode": 0,
        "Error": "",
        "StartedAt": "2022-04-27T10:16:33.16213946Z",
        "FinishedAt": "2001-01-01T00:00:00Z"
    },
    "Image": "sha256:20106db9aa7ac731551a17d6ec07e61fb442221460e5106fa70be71d98b5ee85",
    "ResolvConfPath": "/var/lib/docker/containers/13b9509ccc2fc5f0a2f77a58d7b3f247c4160062d02a7356ce3580637407aa7b/resolv.conf",
    "HostnamePath": "/var/lib/docker/containers/13b9509ccc2fc5f0a2f77a58d7b3f247c4160062d02a7356ce3580637407aa7b/hostname",
    "HostsPath": "/var/lib/docker/containers/13b9509ccc2fc5f0a2f77a58d7b3f247c4160062d02a7356ce3580637407aa7b/hosts"
},
```

7. Um einen neuen Namen für Docker zu geben:

```
# docker run -it --name bash ubuntu ubuntu
```

```
C:\Users\Selami Demiral>docker run -it --name bash_ubuntu ubuntu  
root@4b0995a299a0:/#
```

```
C:\ Eingabeaufforderung
C:\Users\Selami Demiral>docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
4b0995a299a0 ubuntu "bash" 49 seconds ago Up 49 seconds 0.0.0.0:80->80/tcp bash_ubuntu
27078ad5c95e docker/getting-started "/docker-entrypoint...." 2 hours ago Up 2 hours tender_sinoussi

C:\Users\Selami Demiral>
```

Mit diesem Befehl können wir den Container nun kurz ausführen.

```
# docker start bash_ubuntu (Docker läuft im Hintergrund weiter)
```

```
# docker stop bash_ubuntu oder # docker stop 4b (CONT: ID)
```

8. Stoppen und Starten eines Containers :

```
Gokhan-MacBook-Pro:~ gkandemir$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
a5f76b3d5607 phpmyadmin/phpmyadmin "/docker-entrypoint..." 3 seconds ago Up 3 seconds 0.0.0.0:8000→80/tcp pmyadmin
1c143flea181 mysql "docker-entrypoint.s..." 2 minutes ago Up 2 minutes 0.0.0.0:3306→3306/tcp, 33060/tcp mysql-server

Gokhan-MacBook-Pro:~ gkandemir$ docker stop a5f76b3d5607
a5f76b3d5607
Gokhan-MacBook-Pro:~ gkandemir$ docker start pmyadmin
pmyadmin
Gokhan-MacBook-Pro:~ gkandemir$
```

9. Alte laufende Container löschen:

```
# docker rm docker_name oder # docker rm 4b c6 26 93 27 oder
```

```
# docker container rm $(docker container ls -aq)
```

```
C:\Users\Selami Demiral>docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
de23a0784d4f ubuntu "bash" 4 minutes ago Exited (137) 3 minutes ago
4b0995a299a0 ubuntu "bash" 20 minutes ago Exited (137) 9 minutes ago
c60382eb9444 ubuntu "bash" 32 minutes ago Exited (0) 22 minutes ago
26cccbda6473 ubuntu "bash" 39 minutes ago Exited (0) 32 minutes ago
932c72ce0590 ubuntu "sleep 3" 40 minutes ago Exited (0) 40 minutes ago
27078ad5c95e docker/getting-started "/docker-entrypoint..." 2 hours ago Exited (0) 7 minutes ago

C:\Users\Selami Demiral>docker rm angry_franklin
angry_franklin

C:\Users\Selami Demiral>docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
4b0995a299a0 ubuntu "bash" 20 minutes ago Exited (137) 9 minutes ago
c60382eb9444 ubuntu "bash" 32 minutes ago Exited (0) 23 minutes ago
26cccbda6473 ubuntu "bash" 40 minutes ago Exited (0) 33 minutes ago
932c72ce0590 ubuntu "sleep 3" 41 minutes ago Exited (0) 40 minutes ago
27078ad5c95e docker/getting-started "/docker-entrypoint..." 2 hours ago Exited (0) 8 minutes ago

C:\Users\Selami Demiral>docker rm 4b c6 26 93 27
4b
c6
26
93
27

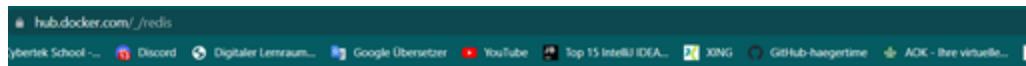
C:\Users\Selami Demiral>docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

C:\Users\Selami Demiral>
```

10. Um gewünschte Docker Version (Tag) zu führen;

```
# docker run redis:5
```

```
c:\Users\Selami Demiral>docker run redis:5
Unable to find image 'redis:5' locally
5: Pulling from library/redis
1fe172e4850f: Already exists
6fbcc347bf99: Already exists
993114c67627: Already exists
990846de095b: Pull complete
e96281c1a982: Pull complete
5a9917af2749: Pull complete
Digest: sha256:a02f17dede9b7592d8e42ae7a8899fd0e62381f27d72dff5c28cba89370054a
Status: Downloaded newer image for redis:5
1:C 26 Apr 2022 10:23:23.323 # <0000:0000:0000: Redis is starting o000:0000:0000
1:C 26 Apr 2022 10:23:23.323 # Redis version=5.0.14, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 26 Apr 2022 10:23:23.323 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
1:M 26 Apr 2022 10:23:23.324 * Running mode=standalone, port=6379.
1:M 26 Apr 2022 10:23:23.324 # Server initialized
1:M 26 Apr 2022 10:23:23.324 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm-overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
1:M 26 Apr 2022 10:23:23.324 # WARNING you have transparent Huge Pages (THP) support enabled in your kernel. This will create latency and memory usage issues with Redis. To fix this issue run the command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retain the setting after a reboot. Redis must be restarted after THP is disabled.
1:M 26 Apr 2022 10:23:23.325 * Ready to accept connections
```



Quick reference

- Maintained by:
the Docker Community
- Where to get help:
the Docker Community Forums, the Docker Community Slack, or Stack Overflow

Supported tags and respective Dockerfile links

- [7.0-rc3](#), [7.0-rc](#), [7.0-rc3-bullseye](#), [7.0-rc-bullseye](#)
- [7.0-rc3-alpine](#), [7.0-rc-alpine](#), [7.0-rc3-alpine3.15](#), [7.0-rc-alpine3.15](#)
- [6.2.6](#), [6.2](#), [6](#), [latest](#), [6.2.6-bullseye](#), [6.2-bullseye](#), [6-bullseye](#), [bullseye](#)
- [6.2.6-alpine](#), [6.2-alpine](#), [6-alpine](#), [alpine](#), [6.2.6-alpine3.15](#), [6.2-alpine3.15](#), [6-alpine3.15](#), [alpine3.15](#)
- [6.0.16](#), [6.0](#), [6.0.16-bullseye](#), [6.0-bullseye](#)
- [6.0.16-alpine](#), [6.0-alpine](#), [6.0.16-alpine3.15](#), [6.0-alpine3.15](#)
- [5.0.14](#), [5.0](#), [5](#), [5.0.14-bullseye](#), [5.0-bullseye](#), [5-bullseye](#)
- [5.0.14-32bit](#), [5.0-32bit](#), [5-32bit](#), [5.0.14-32bit-bullseye](#), [5.0-32bit-bullseye](#), [5-32bit-bullseye](#)
- [5.0.14-alpine](#), [5.0-alpine](#), [5-alpine](#), [5.0.14-alpine3.15](#), [5.0-alpine3.15](#), [5-alpine3.15](#)

```
C:\Users\Selami Demiral>docker images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
mongo              latest   20106db9aa7a  4 days ago   700MB
ubuntu             latest   3f4714ee068a  4 days ago   77.8MB
redis              5        7891e1b96087  5 days ago   110MB
redis              latest   3c3da61c4be0  5 days ago   113MB
docker/getting-started  latest   cb90f98fd791  2 weeks ago  28.8MB
```

C:\Users\Selami Demiral>

11. Um einen eigenen TAG zu vergeben;

```
#docker image tag ubuntu my-ubuntu
```

```
C:\Users\Selami Demiral>docker image tag ubuntu my-ubuntu
```

```
C:\Users\Selami Demiral>docker images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
mongo              latest   20106db9aa7a  4 days ago   700MB
my-ubuntu          latest   3f4714ee068a  4 days ago   77.8MB
ubuntu             latest   3f4714ee068a  4 days ago   77.8MB
redis              5        7891e1b96087  5 days ago   110MB
redis              latest   3c3da61c4be0  5 days ago   113MB
docker/getting-started  latest   cb90f98fd791  2 weeks ago  28.8MB
```

C:\Users\Selami Demiral>

12. Um den Container im Hintergrund auszuführen (Detach mode);

```
# docker run -d redis
```

```
C:\Users\Selami Demiral>docker run -d redis
943a28a05792e7faa72d2b3db27718f28c16170f61c3bedf0715b1eb53743c31
```

```
C:\Users\Selami Demiral>docker ps
CONTAINER ID   IMAGE      COMMAND       CREATED        STATUS        PORTS     NAMES
943a28a05792   redis      "docker-entrypoint.s..."  13 seconds ago   Up 11 seconds  6379/tcp   naughty_boyd
```

Um es wieder in den Vordergrund zu rücken;

```
# docker attach 943a
```

13. So führen Sie einen benutzerdefinierten (privat) Docker aus dem Repository aus;

```
# docker run gkandemir/counter-app
```

14. Um die Logs zu sehen, die wir nicht sehen, während wir im Hintergrund arbeiten;

```
# docker container logs 943a28a05792
```

15. Um eine Benutzeranmeldung vorzunehmen (it : interactive Terminal);

```
# docker run -it gkandemir/interactive-terminal-app
```

```
C:\Users\Selami Demiral>docker pull gkandemir/interactive-terminal-app
Using default tag: latest
latest: Pulling from gkandemir/interactive-terminal-app
8aff230071c9: Already exists
11e9dbdbdef22: Already exists
8033f45d790f: Already exists
4bd891e73716: Already exists
9c666c504964: Already exists
84e8dcfa5f29e: Pull complete
a5062741d567: Pull complete
Digest: sha256:9fddd2eeba28600cb06e376d29d93b1e4a2b03d47162580747a8d1242dda2df8
Status: Downloaded newer image for gkandemir/interactive-terminal-app:latest
docker.io/gkandemir/interactive-terminal-app:latest

C:\Users\Selami Demiral>docker run -it gkandemir/interactive-terminal-app
Heey, Hadi tanışalım. Adın nedir?...: hello
Nerelisin?...: Corum
Kaç Yaşandasın?...: 44
[=====SONUC=====]
hello, Corum, 44
[=====]
Hadi eyvallah!!!

C:\Users\Selami Demiral>
```

C:\Users\USER>docker run -it gkandemir/interactive-terminal-app

16. Auflistung der Netzwerker

```
C:\Users\USER>docker network list
NETWORK ID      NAME      DRIVER      SCOPE
df0b9faa292e    bridge    bridge      local
ecf88c94016d    host      host       local
8d7111665d3b    none      null       local
```

17. Inspektieren eines Netzwerks

```
C:\Users\USER> docker inspect bridge
```

18. Erstellen eines Custom Netzwerks

```
C:\Users\USER>docker network create --driver bridge --subnet 182.18.0.1/24 --gateway 182.18.0.1 custom-network
70c09ad21a680e9cc0dd42e62b434c0e72eb1d9da62fb92a807fb4fdcc6d4fd3
```

```
C:\Users\USER>docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
df0b9faa292e    bridge    bridge      local
70c09ad21a68    custom-network    bridge      local
ecf88c94016d    host      host       local
8d7111665d3b    none      null       local
```

```
C:\Users\USER>
```

Arbeitsarchitektur von Docker



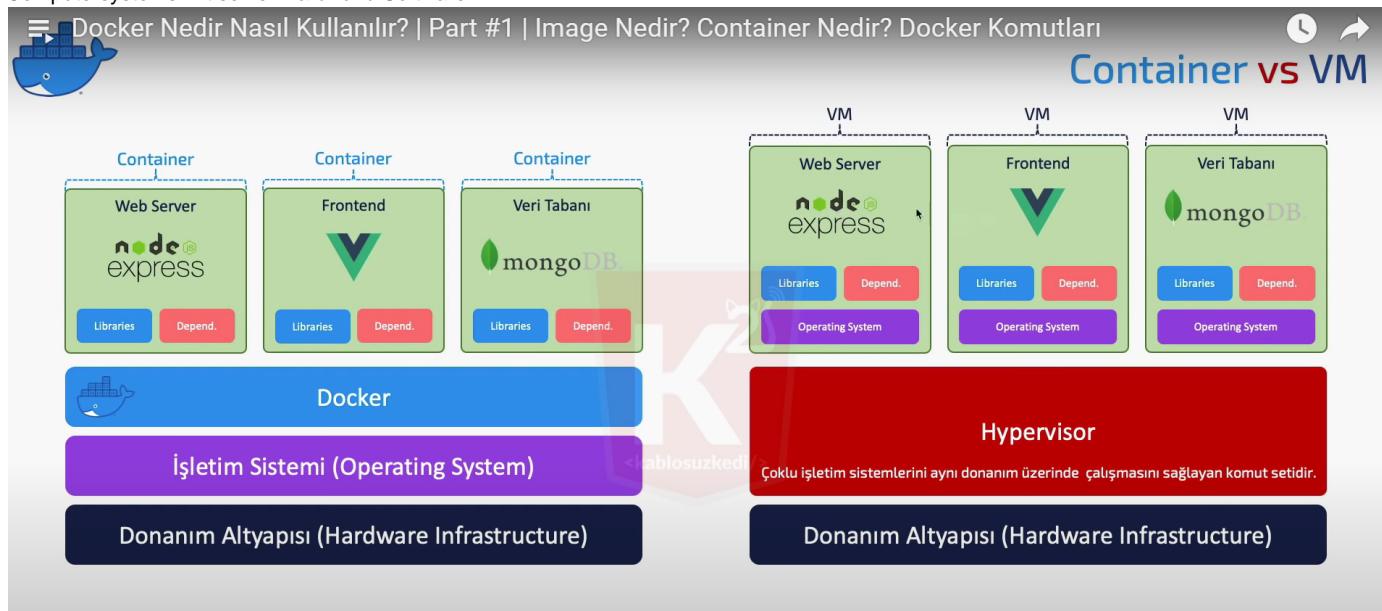
Container vs Virtual Machine Vergleich

Container und Virtuelle Maschinen ähneln sich, was die Isolation und Zuweisung von Ressourcen angeht.

Sie unterscheiden sich aber dadurch, dass Container lediglich Betriebssysteme virtualisieren, keine Hardware. Ein Container läuft auf der echten Hardware eines Hosts und nicht auf einer virtuellen, simulierten Hardware.

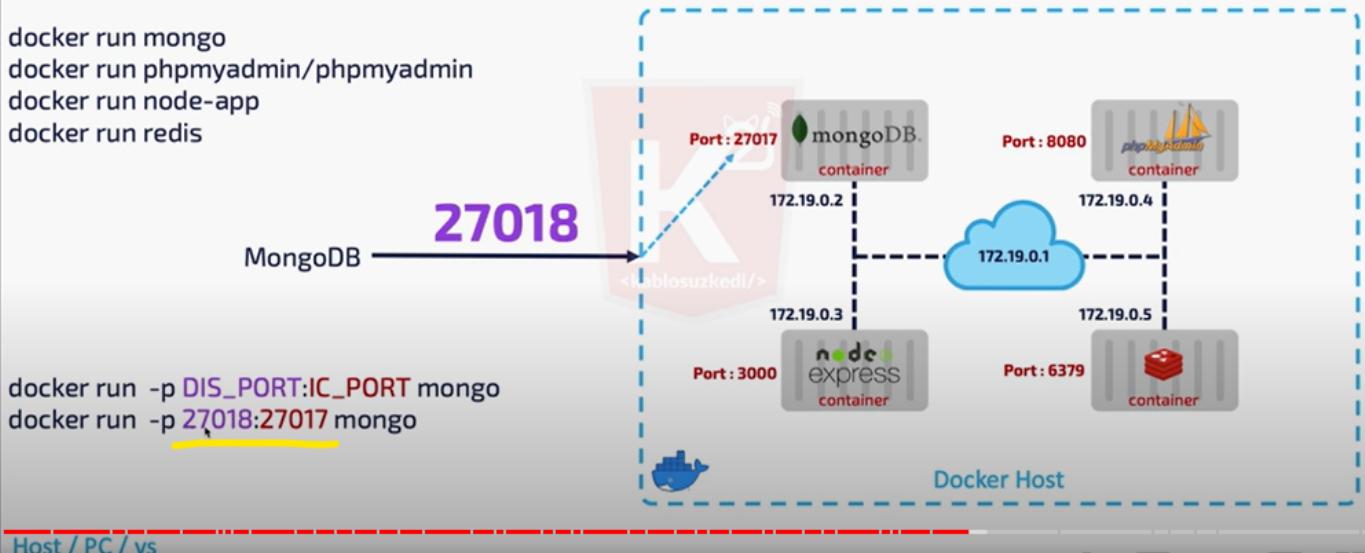
Alle Container werden innerhalb eines einzelnen Betriebssystem Kernels ausgeführt. Das macht sie leichtgewichtiger als Virtuelle Maschinen.

Virtuelle Maschinen emulieren ein komplettes Computersystem mit seiner Architektur und bieten die Funktionalität eines physischen Computersystems mit seiner Hard- und Software.



3. EXPOSING PORTS und PORT MAPPING

Wenn wir ein Container-Image ausführen, läuft dieser Container durch einen inneren Port. Wir können jedoch mehr als einen Container mit unterschiedlichen Außenports durch diesen Container und diesen Innenport führen.



```

C:\Users\Selami Demiral>docker run -p 27017:27017 mongo
{"t": {"$date": "2022-04-26T11:10:17.519+00:00"}, "s": "I", "c": "CONTROL", "id": 23285, "ctx": "-", "msg": "Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t": {"$date": "2022-04-26T11:10:17.522+00:00"}, "s": "I", "c": "NETWORK", "id": 4915701, "ctx": "main", "msg": "Initialized wire specification", "attr": {"spec": {"incomingExternalClient": {"minWireVersion": 0, "maxWireVersion": 13}, "incomingInternalClient": {"minWireVersion": 0, "maxWireVersion": 13}, "outgoing": {"minWireVersion": 0, "maxWireVersion": 13}, "isInternalClient": true}}}
{"t": {"$date": "2022-04-26T11:10:17.539+00:00"}, "s": "W", "c": "ASIO", "id": 22601, "ctx": "main", "msg": "No TransportLayer configured during NetworkInterface startup"}
{"t": {"$date": "2022-04-26T11:10:17.540+00:00"}, "s": "I", "c": "NETWORK", "id": 4648601, "ctx": "main", "msg": "Implicit TCP FastOpen unavailable. If TCP FastOpen is required, set tcpFastOpenServer, tcpFastOpenClient, and tcpFastOpenQueueSize."}
{"t": {"$date": "2022-04-26T11:10:17.550+00:00"}, "s": "W", "c": "ASIO", "id": 22601, "ctx": "main", "msg": "No TransportLayer configured during NetworkInterface startup"}
{"t": {"$date": "2022-04-26T11:10:17.551+00:00"}, "s": "I", "c": "REPL", "id": 5123008, "ctx": "main", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationDonorService", "ns": "config.tenantMigrationDonors"}}
{"t": {"$date": "2022-04-26T11:10:17.551+00:00"}, "s": "I", "c": "REPL", "id": 5123008, "ctx": "main", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationRecipientService", "ns": "config.tenantMigrationRecipients"}}
{"t": {"$date": "2022-04-26T11:10:17.551+00:00"}, "s": "I", "c": "CONTROL", "id": 5945603, "ctx": "main", "msg": "Multi threading initialized"}
{"t": {"$date": "2022-04-26T11:10:17.552+00:00"}, "s": "I", "c": "CONTROL", "id": 4615611, "ctx": "initandlisten", "msg": "MongoDB starting", "attr": {"pid": 1, "port": 27017, "dbPath": "/data/db", "architecture": "64-bit", "host": "913cebd2067e"}}

  
```

Andere Beispiel;

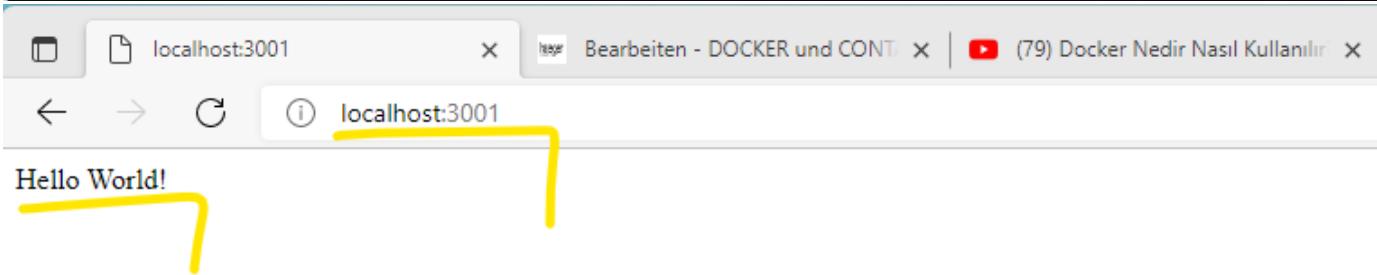
Hier möchten wir 3001 als außer Port verwenden. Innere Port ist schon 3000.

```

C:\Users\Selami Demiral>docker run gkandemir/node-app
Unable to find image 'gkandemir/node-app:latest' locally
latest: Pulling from gkandemir/node-app
f22cccc0b8772: Pull complete
3cf8fb62ba5f: Pull complete
e80c964ece6a: Pull complete
f0d490fd81e4: Pull complete
819364e035c7: Pull complete
24f13810f825: Pull complete
e172be37635d: Pull complete
4b772951db54: Pull complete
7d9e3c496370: Pull complete
Digest: sha256:cf97c85133a6b5451439417db1158627b9b6d647abe5b8e27f7120b7b6c6c076
Status: Downloaded newer image for gkandemir/node-app:latest
Example app listening at http://localhost:3000

C:\Users\Selami Demiral>
  
```

```
C:\Users\Selami Demiral>docker run -p 3001:3000 gkandemir/node-app  
Example app listening at http://localhost:3000
```



Andere Beispiel: Wir können mehrere Container auf demselben inneren Port betreiben.

```
C:\Users\Selami Demiral>docker run -p 3000:3000 -d gkandemir/node-app  
d822bfe4ff4326f13dc3bf251b034c063a6fe48299baa80b99e5941ee7687f05  
  
C:\Users\Selami Demiral>docker run -p 3001:3000 -d gkandemir/node-app  
2ba79fe6eaa66ae0128e0fefef543fb4db478bf4575c68b850818d9fb467d42c  
  
C:\Users\Selami Demiral>docker run -p 3002:3000 -d gkandemir/node-app  
2bddc0935888036904113f188b1c246e9f56d1863ce6468dbbbaba64fb1fd4d9  
  
C:\Users\Selami Demiral>docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
2bddc0935888 gkandemir/node-app "node app.js" 6 seconds ago Up 5 seconds 0.0.0.0:3002->3000/tcp quirky_nash  
2ba79fe6eaa6 gkandemir/node-app "node app.js" 12 seconds ago Up 11 seconds 0.0.0.0:3001->3000/tcp peaceful_lamport  
d822bfe4ff43 gkandemir/node-app "node app.js" 18 seconds ago Up 18 seconds 0.0.0.0:3000->3000/tcp hopeful_almeida  
  
C:\Users\Selami Demiral>
```

oder alle Ports in einer Zeile:

```
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM.exe Dosyaları\DOCKER\RESOURCES>docker run -d -p 3000:3000 -p 3001:3000 -p 3002:3000 gkandemir/node-app  
737867cdf8bec58bcff213784348cd122316495c8f50356756c02893b4cfbb1  
  
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM.exe Dosyaları\DOCKER\RESOURCES>docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
737867cdf8be gkandemir/node-app "node app.js" 6 seconds ago Up 6 seconds 0.0.0.0:3000->3000/tcp, 0.0.0.0:3001->3000/tcp, 0.0.0.0:3002->3000/tcp heuristic Ritchie  
  
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM.exe Dosyaları\DOCKER\RESOURCES>
```

Erstellen ein Tag:

```
C:\Users\USER>docker image tag ubuntu my-ubuntu  
  
C:\Users\USER>docker images  
REPOSITORY TAG IMAGE ID CREATED SIZE  
mongo latest 20106db9aa7a 4 days ago 700MB  
ubuntu latest 3f4714ee068a 4 days ago 77.8MB  
my-ubuntu latest 3f4714ee068a 4 days ago 77.8MB  
redis 5 7891e1b96087 5 days ago 110MB  
redis latest 3c3da61c4be0 5 days ago 113MB  
  
C:\Users\USER>
```

4. VOLUME MAPPING und VERKNÜPFUNG CONTAINERS

- Volumes zwischen Host and Container
- Volumes zwischen Containers

Container laufen STATELESS auf Docker Host, daher werden keine Informationen darin aufgezeichnet. Wenn der Container gestoppt wird, werden die von uns aufgezeichneten Informationen für immer gelöscht.

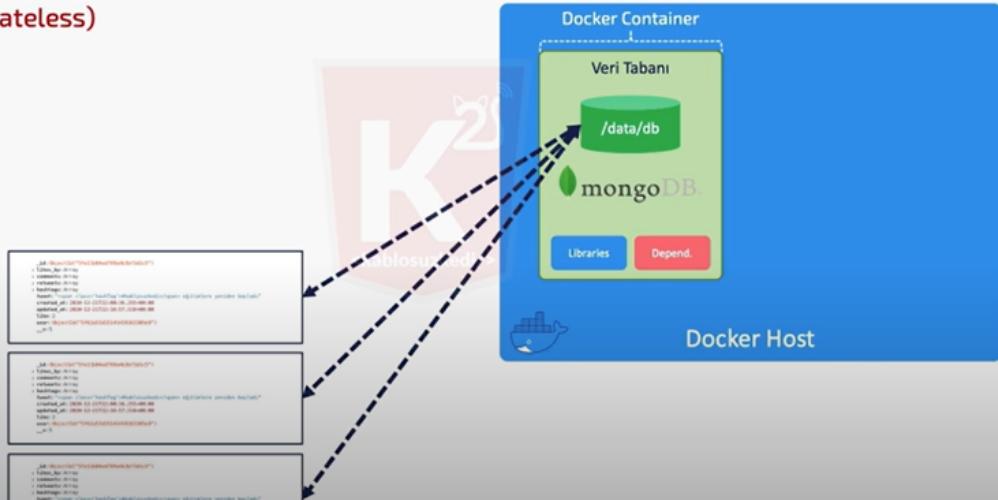
Aber der Lebenszyklus unserer Anwendungen ist nicht genau so! In Docker wird Volume verwendet, um diese Situation zu beseitigen.

Dazu geben wir beim Booten eines Containers die Adresse des Ordners auf dem Docker-Host an, der gespeichert werden soll es und überträgt es in den Ordner, den wir gemappt haben. Nachdem der Container gestoppt wurde, verbleiben die Informationen auf dem Docker-Host.



Volume Mapping

docker run mongo (**stateless**)



Volume Mapping

docker run mongo (**stateless**)

docker stop mongo

docker run mongo (**stateless**)

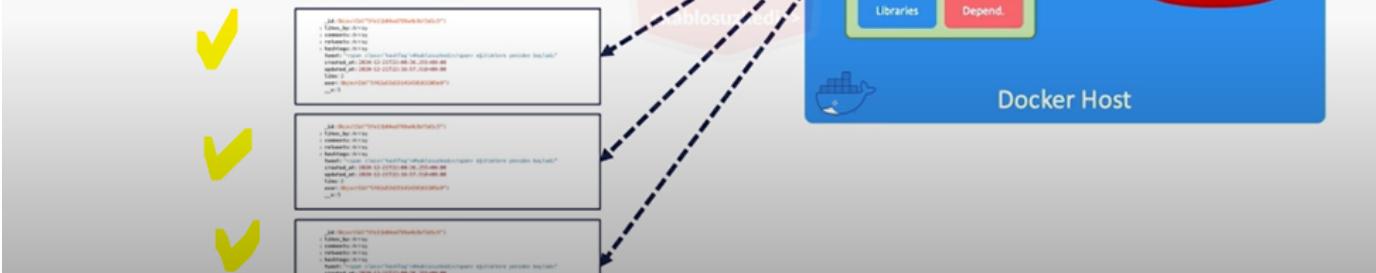


Vorherige Informationen sind weg. Um dies zu verhindern;

```

docker run mongo (stateless)
docker stop mongo
docker run mongo (stateless)
docker stop mongo
docker run -v /opt/data:/data/db mongo
docker stop mongo
docker run -v /opt/data:/data/db mongo

```



Beispiel für Volume Mapping;

Ausführen MySql von localhost :

MySQL Beispiel ;

```

# docker pull mysql
# docker run -e MYSQL_ROOT_PASSWORD=test123 -d mysql

```

MySQL läuft als Host (Drinnen):

```
C:\Users\Selami Demiral>docker ps
CONTAINER ID   IMAGE      COMMAND          CREATED        STATUS        PORTS          NAMES
6756907ce2db   mysql      "docker-entrypoint.s..."  2 minutes ago  Up 2 minutes  3306/tcp, 33060/tcp   serene_haw
king
13b9509ccc2f   mongo      "docker-entrypoint.s..."  11 minutes ago Up 11 minutes  0.0.0.0:27017->27017/tcp   vibrant_ch
andrasekhar
```

Wir können diese Anwendungen stoppen.

Nun, zuerst führen wir den Container aus, um eine Verbindung zu ihm herzustellen;

```
# docker run --name mysql-server -p 3306:3306 -e MYSQL_ROOT_PASSWORD=test123 -d mysql
```

```
C:\Users\Selami Demiral>docker run --name mysql-server -p 3306:3306 -e MYSQL_ROOT_PASSWORD=test123 -d mysql
a9e27a4ef25a9a025426da7449c12b5eebbcacf4e3df98cb6a30fcba895efc52
docker: Error response from daemon: Ports are not available: listen tcp 0.0.0.0:3306: bind: Only one usage of each socket address (protocol/network address/port) is normally permitted.
C:\Users\Selami Demiral>
```

Wenn ein solcher Fehler empfangen wird,

Sie können das Problem lösen, indem Sie den Prozess „mysqld.exe“ im **Task-Manager** beenden.

```
C:\Users\Selami Demiral>docker run --name mysql-server -p 3306:3306 -e MYSQL_ROOT_PASSWORD=test123 -d mysql
docker: Error response from daemon: Conflict. The container name "/mysql-server" is already in use by container "a9e27a4
ef25a9a025426da7449c12b5eebbcacf4e3df98cb6a30fcba895efc52". You have to remove (or rename) that container to be able to
reuse that name.
See 'docker run --help'.
C:\Users\Selami Demiral>
```

Wenn ein solcher Fehler empfangen wird, können wir laufende mysql Container wie unten sehen.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS	NAMES			
d5cbbc680d2f	mysql	"docker-entrypoint.s..."	11 minutes ago	Exited (1) 11 minutes ago
a9e27a4ef25a	mysql	"docker-entrypoint.s..."	4 hours ago	Created
mysql-server				
6756907ce2db	mysql	"docker-entrypoint.s..."	4 hours ago	Exited (0) 4 hours ago
serene_hawking				

Und wir müssen diese Container löschen.

```
# docker rm a9e
```

Nun können wir den Code laufen lassen;

```
C:\Users\Selami Demiral>docker run --name mysql-server -p 3306:3306 -e MYSQL_ROOT_PASSWORD=test123 -d mysql
f1186178212ea754c95649f4b3d4cc0afdf5268ea2984433559c9f83242a3c142
```

```
C:\Users\Selami Demiral>docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
AMES
f1186178212e mysql "docker-entrypoint.s..." 15 seconds ago Up 13 seconds 0.0.0.0:3306->3306/tcp, 33060/tcp
mysql-server

C:\Users\Selami Demiral>
```

Der anschließende Container ist nun bereit. Jetzt bereiten wir den Container vor, der eine Verbindung herstellt. Dafür nutzen wir den Container im phpmyadmin ;

```
# docker pull phpmyadmin/phpmyadmin
```

Verknüpfung der Containers :

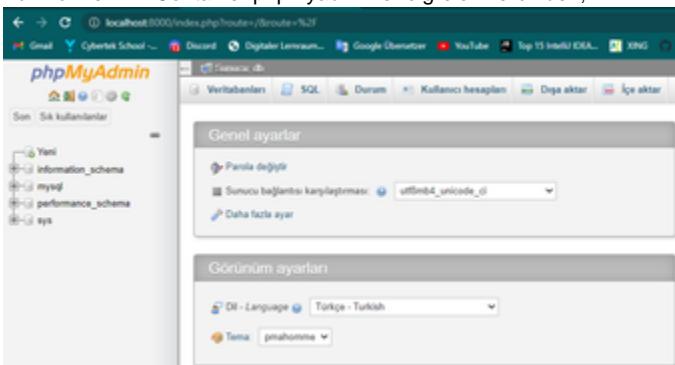
```
# docker run --name pmyadmin -p 8000:80 --link mysql-server:db -d phpmyadmin/phpmyadmin
```

Da MySQL als DB in phpmyadmin eingebunden ist, schreiben wir **db** als Alias.

```
C:\Users\Selami Demiral>docker run --name pmyadmin -p 8000:80 --link mysql-server:db -d phpmyadmin/phpmyadmin
e116c1de3231714062f745825b5aa88b5e91c3607d3a6bed48123f23e0887bc8
```

```
C:\Users\Selami Demiral>docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
e116c1de3231 phpmyadmin/phpmyadmin "/docker-entrypoint..." 6 seconds ago Up 5 seconds 0.0.0.0:8000->80/tcp pmyadmin
f1186178212e mysql "docker-entrypoint.s..." 6 minutes ago Up 6 minutes 0.0.0.0:3306->3306/tcp, 33060/tcp mysql-server
```

Nun können wir Container phpmyadmin erfolgreich verbinden;



Da wir keine Volume Mapping gemacht habe, werden alle Dateien gelöscht, wenn wir den Container phpmyadmin stoppen. Las uns **Volume Mapping** machen;

```
C:\Users\Selami Demiral>docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
e116c1de3231 phpmyadmin/phpmyadmin "/docker-entrypoint..." 10 minutes ago Up 10 minutes 0.0.0.0:8000->80/tcp pmyadmin
f1186178212e mysql "docker-entrypoint.s..." 17 minutes ago Up 17 minutes 0.0.0.0:3306->3306/tcp, 33060/tcp mysql-server

C:\Users\Selami Demiral>docker stop e1 f1
e1
f1

C:\Users\Selami Demiral>docker rm e1 f1
e1
f1

C:\Users\Selami Demiral>
```

The `docker exec` command allows you to run commands inside a Docker container. The following command line will:

```
$ docker exec -it some-mysql bash
```

The log is available through Docker's container log:

```
$ docker logs some-mysql
```

Using a custom MySQL configuration file

The default configuration for MySQL can be found in `/etc/mysql/my.cnf`, which may include additional directives. Please inspect the relevant files and directories within the `mysql` image itself for more details.

If `/my/custom/config-file.cnf` is the path and name of your custom configuration file, you can start your `mysql` container using:

```
$ docker run --name some-mysql -v /my/custom:/etc/mysql/conf.d -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql:tag
```

This will start a new container `some-mysql`, where the MySQL instance uses the combined startup settings from `/etc/my.cnf` and `/my/custom/config-file.cnf`, with the latter taking precedence.

```
# docker run --name mysql-server -p 3306:3306 -v /opt/data/etc/mysql/conf.d -e MYSQL_ROOT_PASSWORD=test123 -d mysql
C:\Users\Selami Demiral>docker run --name mysql-server -p 3306:3306 -v /opt/data/etc/mysql/conf.d -e MYSQL_ROOT_PASSWORD=test123 -d mysql
736256c8a9757116ccc2cfedfb0d98bb01684ab0a83f1d30f04ba539a858abb

C:\Users\Selami Demiral>docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
736256c8a975 mysql "docker-entrypoint.s..." 7 seconds ago Up 6 seconds 0.0.0.0:3306->3306/tcp, 33060/tcp mysql-server

C:\Users\Selami Demiral>

C:\Users\Selami Demiral>docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
736256c8a975 mysql "docker-entrypoint.s..." 7 seconds ago Up 6 seconds 0.0.0.0:3306->3306/tcp, 33060/tcp mysql-server

C:\Users\Selami Demiral>docker stop 73
73

C:\Users\Selami Demiral>docker start mysql-server
mysql-server

C:\Users\Selami Demiral>docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
736256c8a975 mysql "docker-entrypoint.s..." 2 minutes ago Up 5 seconds 0.0.0.0:3306->3306/tcp, 33060/tcp mysql-server

C:\Users\Selami Demiral>docker run --name pmyadmin -p 8000:80 --link mysql-server:db -d phpmyadmin/phpmyadmin
59c88c4fb9675bea520e8ece8f73d2b5eec90dc8cf2f8f200f931a44f90a80e

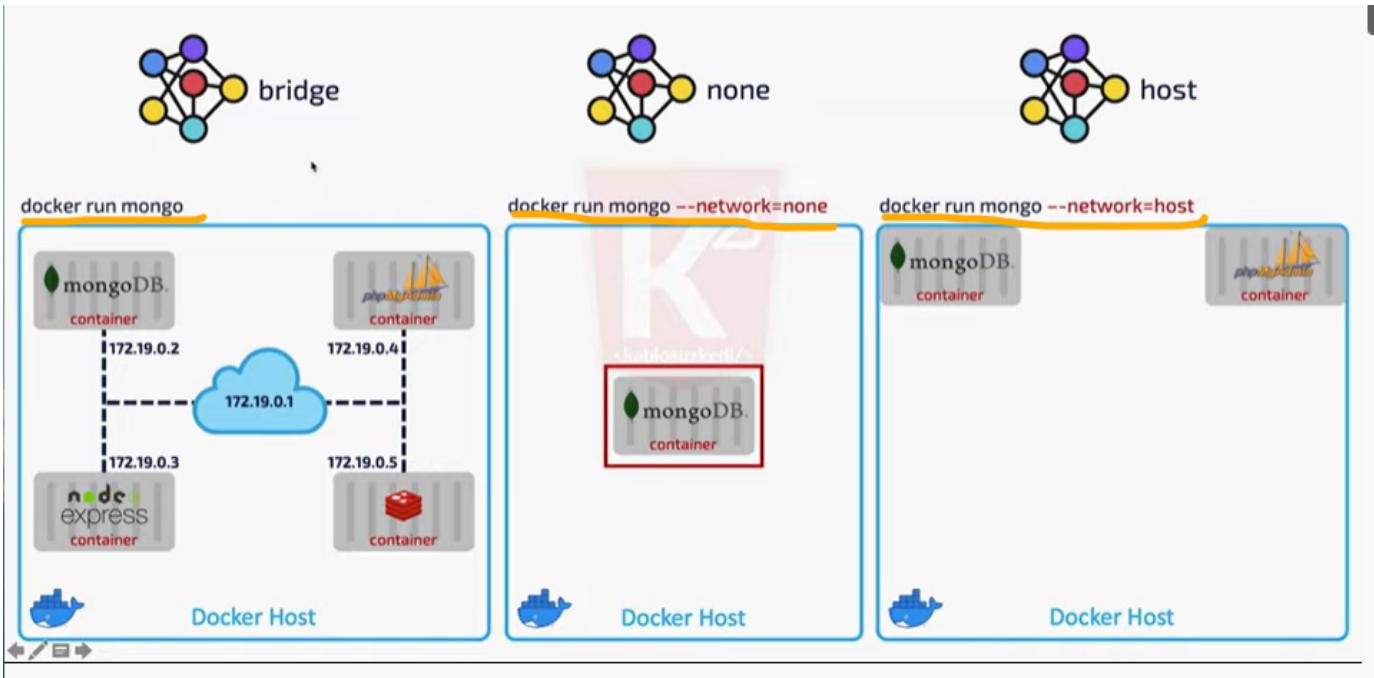
C:\Users\Selami Demiral>docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
59c88c4fb967 phpmyadmin/phpmyadmin "/docker-entrypoint.s..." 12 seconds ago Up 10 seconds 0.0.0.0:8000->80/tcp pmyadmin
736256c8a975 mysql "docker-entrypoint.s..." 3 minutes ago Up 53 seconds 0.0.0.0:3306->3306/tcp, 33060/tcp mysql-server
```

Jetzt können wir den Container öffnen und schließen, ohne die von uns erstellten Tabelleninformationen zu löschen. Wir haben Volume Mapping erstellt.

6. DOCKER NETZWERKTYPEN

Der Container Host stellt seinen Docker Containern Netzwerke zur Verfügung, über die Container miteinander oder mit Client-Anwendungen kommunizieren können.

- Bridge Network
- None Network
- Host Network
- Benutzerdefiniertes Network

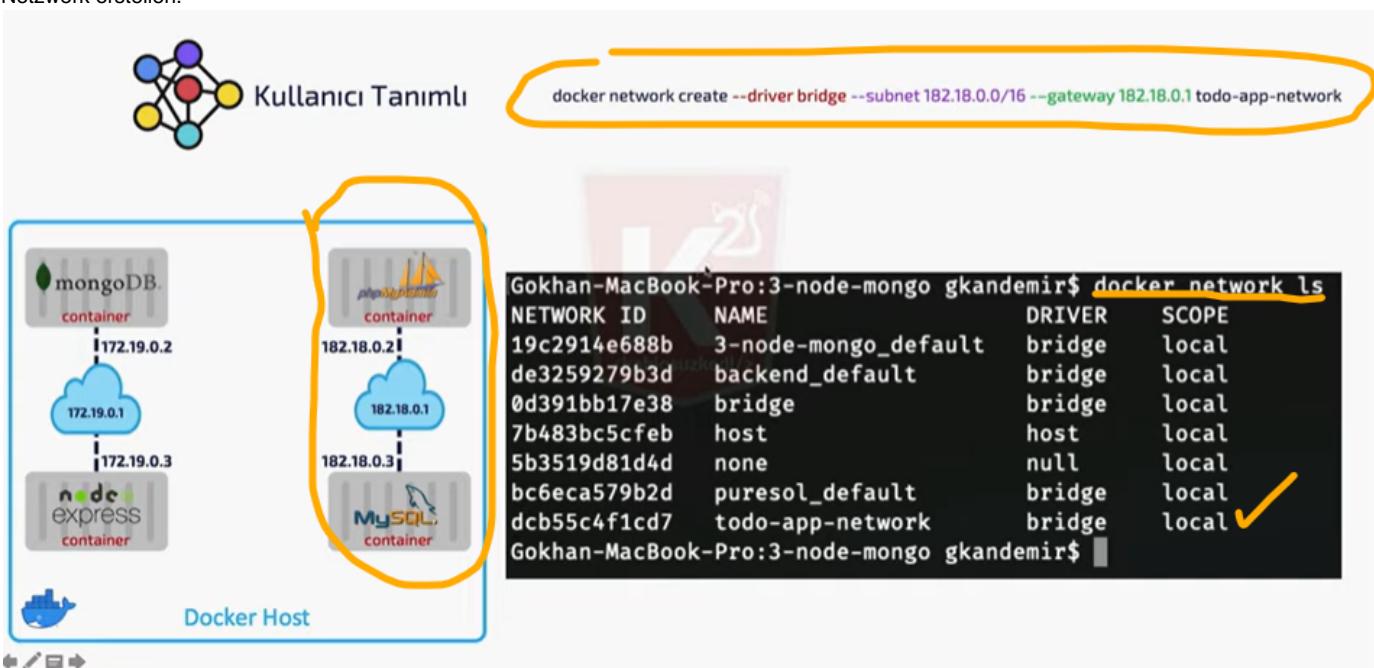


Bridge Network :Docker Host hat ein Gateway und andere Container haben ihre eigene IP und Ports durch dieses Gateway. Und Container sind über dieses Gateway miteinander verbunden. Das standardmäßig verwendete Netzwerk ist Bridge Network.

Network None :Wenn wir nicht möchten, dass von außen auf einen Container zugegriffen wird, verwenden wir Network None.

Network Host :Der Host hat eine eigene IP. Docker erhält nur Portinformationen vom Host und ermöglicht uns den Zugriff auf diese Container

Benutzerdefiniertes Network :Normalerweise haben wir standardmäßig ein Bridge-Netzwerk. Wir können jedoch auch ein benutzerdefiniertes Netzwerk erstellen.



1. Entfernung Images;
docker rmi my-ubuntu
2. Netzwerke Auflisten ;
docker network ls oder # docker network list
3. Netzwerk löschen;
docker network rm my-network

```

Gokhan-MacBook-Pro:~ gkandemir$ docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
c227dbb29377   bridge    bridge      local
7b483bc5cfab  host      host       local
71e5ae8de3c3  my-network  bridge      local
5b3519d81d4d  none      null      local
Gokhan-MacBook-Pro:~ gkandemir$ docker network list
NETWORK ID      NAME      DRIVER      SCOPE
c227dbb29377   bridge    bridge      local
7b483bc5cfab  host      host       local
71e5ae8de3c3  my-network  bridge      local
5b3519d81d4d  none      null      local
Gokhan-MacBook-Pro:~ gkandemir$ docker network rm my-network
my-network
Gokhan-MacBook-Pro:~ gkandemir$ --lşnkt

```

Network Beispiel ;

```

app.listen(PORT, async () => {
  console.log("Sunucu çalışıyor... MongoDB'ye bağlanılacak..");
  // await Mongoose.connect("mongodb://in-address:27017/todos", {
  * await Mongoose.connect("mongodb://mongo-server:27017/todos", {

```

Aufgabe : Wir verbinden Container “todo-app” mit Container “mongo-server” auf einen neuen Netzwerk

docker pull gkandemir/todo-app

1. Zuerst erstellen wir einen Netzwerk, über default “bridge“ Netzwerk;

Man kann subnet und gateway vom eigenen local netzwerk verwenden. Dafür;

ipconfig

```

Drahtlos-LAN-Adapter WLAN:
  Verbindungsspezifisches DNS-Suffix: speedport.ip
  IPv6-Adresse. . . . . : 2003:e2:af26:dc83:f9c1:2d97:4151:aa73
  Temporäre IPv6-Adresse. . . . . : 2003:e2:af26:dc83:8d92:70a5:7eb9:cfd3
  Verbindungslokale IPv6-Adresse . . : fe80::f9c1:2d97:4151:aa73%3
  IPv4-Adresse . . . . . : 192.168.2.148
  Subnetzmaske . . . . . : 255.255.255.0
  Standardgateway . . . . . : fe80::1%3
                                192.168.2.1

Ethernet-Adapter Bluetooth-Netzwerkverbindung:
  Medienstatus. . . . . . . . . : Medium getrennt
  Verbindungsspezifisches DNS-Suffix:

Ethernet-Adapter vEthernet (WSL):
  Verbindungsspezifisches DNS-Suffix:
  Verbindungslokale IPv6-Adresse . . : fe80::ccc1:cdf2:3med:8a0a%39
  IPv4-Adresse . . . . . : 172.23.176.1
  Subnetzmaske . . . . . . . . . : 255.255.240.0
  Standardgateway . . . . . . . . : 
```

docker network create --driver bridge --subnet 172.23.176.1/24 --gateway 172.23.176.1 custom-network

```
C:\Users\Selami Demiral>docker network create --driver bridge --subnet 172.23.176.1/24 --gateway 172.23.176.1 custom-network
bfa39e67d5049c17b8217aed92bb59efc665a27c6aa775d191e1795479a52d8f

C:\Users\Selami Demiral>docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
212644deb9a8   bridge    bridge      local
bfa39e67d504   custom-network  bridge      local
0b6950ba8d76   host      host      local
845760f4e1a0   none      null      local

C:\Users\Selami Demiral>docker inspect bf
[

{
  "Name": "custom-network",
  "Id": "bfa39e67d5049c17b8217aed92bb59efc665a27c6aa775d191e1795479a52d8f",
  "Created": "2022-04-28T10:23:51.068040011Z",
  "Scope": "local",
  "Driver": "bridge",
  "EnableIPv6": false,
  "IPAM": {
    "Driver": "default",
    "Options": {},
    "Config": [
      {
        "Subnet": "172.23.176.1/24",
        "Gateway": "172.23.176.1"
      }
    ]
  }
}
```

2. Jetzt erstellen wir einen Container genannt “mongo-server“ über “custom-network“, den wir gerade erstellt habe.

```
# docker run --name mongo-server --net custom-network -d mongo
C:\Users\Selami Demiral>docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
212644deb9a8   bridge    bridge      local
bfa39e67d504   custom-network  bridge      local
0b6950ba8d76   host      host      local
845760f4e1a0   none      null      local

C:\Users\Selami Demiral>docker ps
CONTAINER ID      IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES

C:\Users\Selami Demiral>docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
mongo          latest    20106db9aa7a  6 days ago   700MB
ubuntu          latest    5f4714ee068a  6 days ago   77.8MB
redis           5         7891e1b96087  7 days ago   100MB
redis           latest    3c3da61c4be0  7 days ago   115MB
mysql           latest    f2ad9ff23df82  8 days ago   521MB
docker/getting-started  latest    cb90f98fd791  2 weeks ago  28.8MB
phpmyadmin/phpmyadmin  latest    5682e7556577  2 months ago  524MB
gkandemir/todo-app  latest    ec5c47421d25  14 months ago  181MB
gkandemir/interactive-terminal-app  latest    cc9ff3f96d0b  14 months ago  167MB
gkandemir/counter-app  latest    dc6eb63b9bfc  14 months ago  167MB
gkandemir/node-app  latest    917080a94ca3  15 months ago  238MB

C:\Users\Selami Demiral>docker run --name mongo-server --net custom-network -d mongo
e8fc5a45a154c99d4bed6eb417b1ba4fdd2eb11ca5d4c8a596ba664ef8937290

C:\Users\Selami Demiral>docker ps
CONTAINER ID      IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
e8fc5a45a154      mongo      "docker-entrypoint.s..."  18 seconds ago  Up 17 seconds  27017/tcp  mongo-server
```

3. Einschließlich laufen wir den bestehende Container “todo-app“ über custom-network

```
# docker run --net custom-network -p 3000:3000 gkandemir/todo-app
C:\Users\Selami Demiral>docker run --net custom-network -p 3000:3000 gkandemir/todo-app
Sunucu çalışıyor... MongoDB'ye bağlanılacak..
MongoDB'ye bağlantı başarılı!
```

4. Nun haben wir zwei Containers, die über custom-network laufen

```
C:\Users\Selami Demiral>docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
604211b566a3 gkandemir/todo-app "docker-entrypoint.s..." 11 minutes ago Up 11 minutes 0.0.0.0:3000->3000/tcp zen_spence
e8fc5a45a154 mongo "docker-entrypoint.s..." 28 minutes ago Up 28 minutes 27017/tcp mongo-server

C:\Users\Selami Demiral>docker network ls
NETWORK ID NAME DRIVER SCOPE
212644deb9a8 bridge bridge local
bfa39e67d504 custom-network bridge local
0b6950ba8d76 host host local
845760f4e1a0 none null local

C:\Users\Selami Demiral>docker inspect bf
{
    "Subnet": "172.23.176.1/24",
    "Gateway": "172.23.176.1"
}
],
"Internal": false,
"Attachable": false,
"Ingress": false,
"ConfigFrom": {
    "Network": ""
},
"ConfigOnly": false,
"Containers": [
    "604211b566a38d9d4fcf14802ab7fed8ee0e27e0c23dbef9f40ebf2e37ef38ab": {
        "Name": "zen_spence",
        "EndpointID": "8070d8eb12ad040ab075b5a7bdb288fbacfba9b59642f785a3ec8fa7a9c6c208",
        "MacAddress": "02:42:ac:17:b0:03",
        "IPv4Address": "172.23.176.3/24",
        "IPv6Address": ""
    },
    "e8fc5a45a154c99d4bed6eb417b1ba4fdd2eb11ca5d4c8a596ba664ef8937290": {
        "Name": "mongo-server",
        "EndpointID": "e993a246c41e356f4c0dfdb74a788f34795e6193f1c56c032398eee9b90ee6b2",
        "MacAddress": "02:42:ac:17:b0:02",
        "IPv4Address": "172.23.176.2/24",
        "IPv6Address": ""
    }
]
}
```

5. Jetzt können wir einen der Container auf postman ausführen und testen.

The screenshot shows the Postman application interface. On the left, there's a sidebar with sections for Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The main workspace is titled 'My Workspace' and contains a collection named 'Cypress Api Testing'. Inside this collection, there are several requests: 'GET New Request', 'haegertime', 'haegertimeTestsAPI_AdminUser', 'haegertimeTestsAPI_BookkeeperUser', 'haegertimeTestsAPI_EmployeeUser', and 'New Collection'. A new request is being created for 'POST' to 'http://localhost:3000/todo'. The 'Body' tab is selected, showing a JSON payload:

```
{
  "title": "title4",
  "description": "description 4",
  "completed": true
}
```

Below the body, the 'Test Results' section shows a successful response with a status of '200 OK', a duration of '16 ms', and a size of '414 B'. The response body is also displayed in the 'Pretty' format:

```

1 _id: "626a95b5a857c6e9c9e38612",
2   title: "title4",
3   description: "description 4",
4   completed: true,
5   created_at: "2022-04-28T13:25:09.900Z",
6

```

Und

The screenshot shows a POSTMAN interface with the following details:

- Method:** GET
- URL:** http://localhost:3000
- Body:** JSON (selected)
- Response Status:** 200 OK
- Response Time:** 36 ms
- Response Size:** 859 B

The request body is defined as:

```
1 {  
2   "title": "title4",  
3   "description": "description 4",  
4   "completed": true  
5 }
```

The response body is a list of three documents:11 {
12 "_id": "626a957c4a857c626abe38610",
13 "title": "title2",
14 "description": "description 2",
15 "completed": true,
16 "created_at": "2022-04-28T13:24:04.744Z",
17 "__v": 0
18 },
19 {
20 "_id": "626a9592a857c62402e38611",
21 "title": "title3",
22 "description": "description 3",
23 "completed": true,
24 "created_at": "2022-04-28T13:24:34.176Z",
25 "__v": 0
26 },
27 {
28 "_id": "626a9592a857c62402e38612",
29 }

6. ERSTELLEN UNSERES EIGENEN DOCKER DATEI UND IMAGES

Dazu folgen wir zunächst dem langen Weg. Später werden wir kurz sehen, wie diese Arbeit ausgeführt wird.

```
C:\Users\Selami Demiral>docker pull ubuntu:18.04
18.04: Pulling from library/ubuntu
88736512a147: Pull complete
Digest: sha256:627b1184c9100a22ba9dcf531908b9e24af99fa54e45c10f57852fb890a57ea6
Status: Downloaded newer image for ubuntu:18.04
docker.io/library/ubuntu:18.04

C:\Users\Selami Demiral>docker images
REPOSITORY          TAG      IMAGE ID   CREATED        SIZE
mongo              latest   20106db9aa7a  6 days ago   700MB
ubuntu              latest   3f4714ee068a  6 days ago   77.8MB
ubuntu            18.04  fdf0753c97a9  6 days ago   63.2MB
redis              5        7891e1b96087  8 days ago   110MB
redis              latest   3c3da61c4be0  8 days ago   113MB
mysql              latest   f2ad9f23df82  8 days ago   521MB
docker/getting-started  latest   cb90f98fd791  2 weeks ago  28.8MB
phpmyadmin/phpmyadmin  latest   5682e7556577  2 months ago  524MB
gkandemir/todo-app    latest   ec5c47421d25  14 months ago  181MB
gkandemir/interactive-terminal-app  latest   cc9ff3f96d0b  14 months ago  167MB
gkandemir/counter-app    latest   dc6eb63b9bfc  14 months ago  167MB
gkandemir/node-app     latest   917080a94ca3  15 months ago  238MB

C:\Users\Selami Demiral>docker run -it ubuntu_18.04
Unable to find image 'ubuntu_18.04:latest' locally
docker: Error response from daemon: pull access denied for ubuntu_18.04, repository does not exist or may require 'docker login': denied: requested access to the resource is denied.
See 'docker run --help'.
```

```
C:\Users\Selami Demiral>docker run -it ubuntu:18.04
root@022f5968ae68:/#
```

```
# apt-get update (für ubuntu)
# apt-get install curl -y
```

Wir laden node.js in dieses Betriebssystem (Ubuntu 18.04) herunter und installieren es;

```
# curl -sL https://deb.nodesource.com/setup_10.x | bash
#apt-get install nodejs -y
```

Wir führen alle folgenden Operationen in Ubuntu Container durch;

```
root@022f5968ae68:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@022f5968ae68:/# cd opt
root@022f5968ae68:/opt# ls
root@022f5968ae68:/opt# mkdir node-app
root@022f5968ae68:/opt# cd node-app/
root@022f5968ae68:/opt/node-app# echo 'console.log("nodejsapp from ubuntu ...");' > index.js
root@022f5968ae68:/opt/node-app# ls
index.js
root@022f5968ae68:/opt/node-app# cat index.js
console.log("nodejsapp from ubuntu ...");
root@022f5968ae68:/opt/node-app# node index.js
nodejsapp from ubuntu ...
root@022f5968ae68:/opt/node-app#
root@022f5968ae68:/opt/node-app#
root@022f5968ae68:/opt/node-app# history
 1  apt-get update
 2  apt-get install curl -y
 3  curl -sL https://deb.nodesource.com/setup_10.x | bash
 4  apt-get install nodejs -y
 5  ls
 6  cd opt
 7  ls
 8  mkdir node-app
 9  cd node-app/
10  echo 'console.log("nodejsapp from ubuntu ...");' > index.js
11  ls
12  cat index.js
13  node index.js
14  history
root@022f5968ae68:/opt/node-app#
```



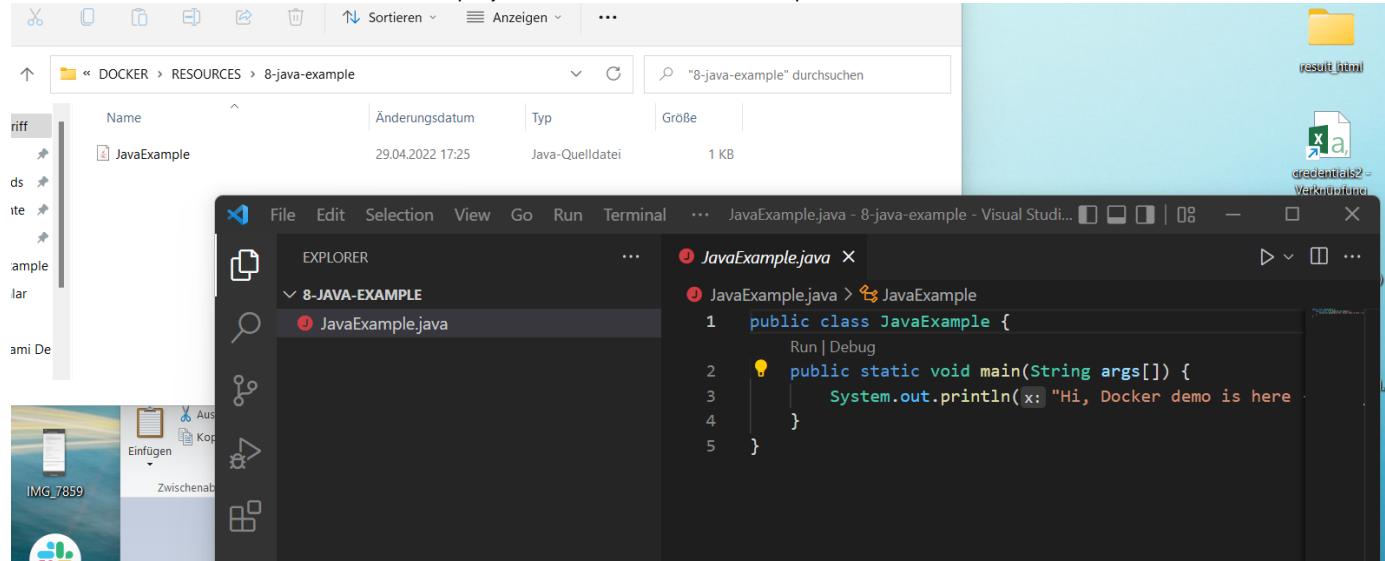
In den nächsten Schritten werden wir jedoch ein Image erstellen, der diese Prozesse automatisiert (Wir herstellen dieses Image um nodejs zu nutzen).

Aufgabe 1 (Haupt Aufgabe) -

Tutorial Link : [How to Create Docker Image for Java Application](#) || [How to Dockerize an application](#) || [Docker Container - YouTube](#)

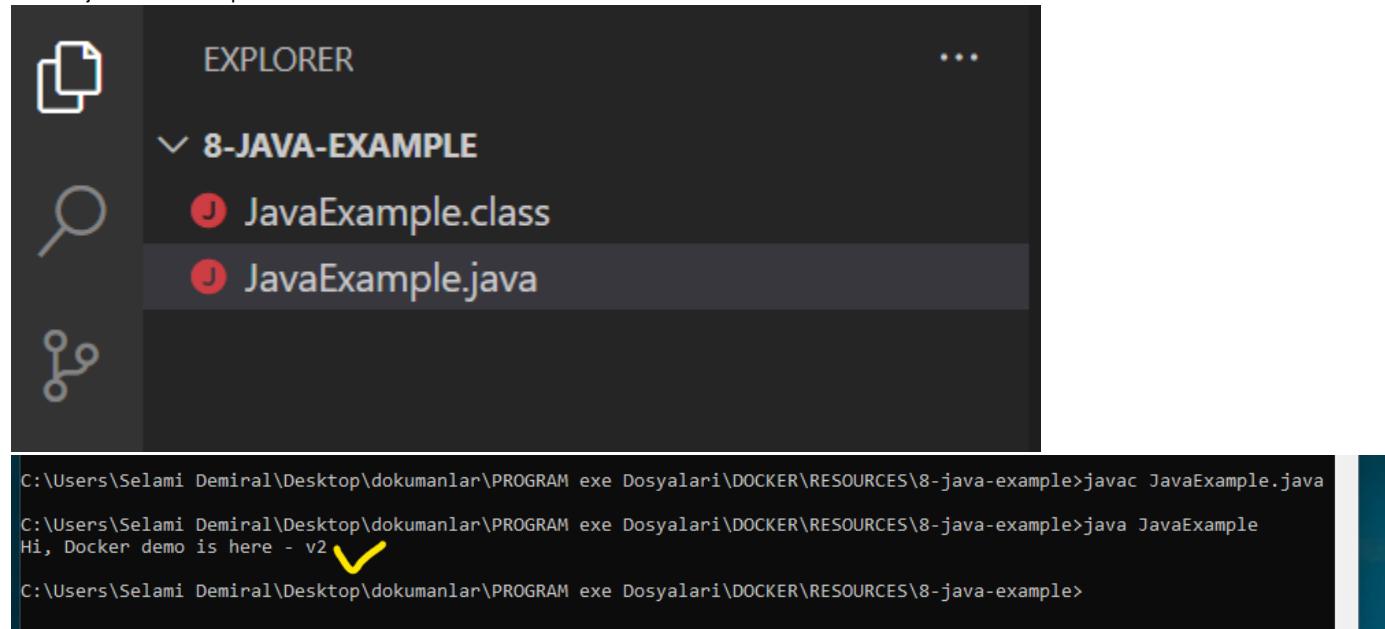
Erstellen Sie eine Java Programm, Docker File, bauen Docker Image & laufen Sie Docker Container für Java Programm.

Zuerst erstellen Sie eine Java Datei "JavaExample.java" und schreiben Sie den simpel Code

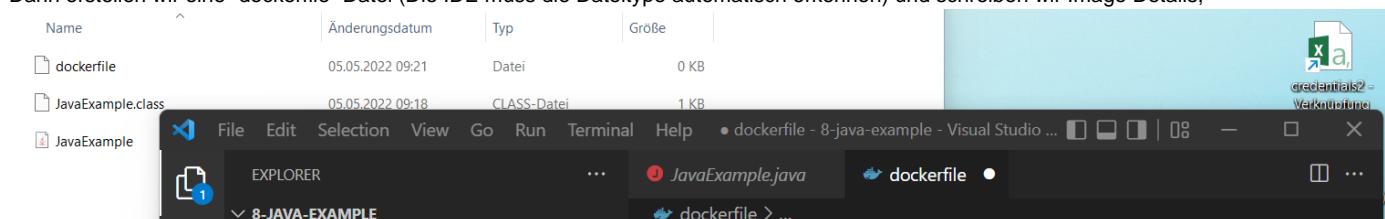


Wenn wir die Datei JavaExample compailen, entstehen sich eine JavaExample.class Datei automatisch. Und wenn wir die datei JavaExample laufen lassen, funktioniert es mit dem Ergebnis.

```
# javac JavaExample.java  
# java JavaExample
```



Dann erstellen wir eine "dockerfile" Datei (Die IDE muss die Dateitype automatisch erkennen) und schreiben wir Image Details;



```

1 #base image details
2 FROM openjdk:8
3 COPY . /src/java
4 WORKDIR /src/java
5 RUN ["javac", "JavaExample.java"]
6 ENTRYPOINT ["java", "JavaExample"]
7

```

Nicht vergessen um Docker Desktop zu laufen lassen und danach laufen Sie das "build" Befehl ;

```

C:\Users\dpgupta\Documents>docker build -t my-java-app:v1 .
Sending build context to Docker daemon 4.096kB
Step 1/5 : FROM openjdk:8
--> 6cedeef72886
Step 2/5 : COPY . /src/java
--> 9c77a03073994
Step 3/5 : WORKDIR /src/java
--> Running in 76499115ccf8
Removing intermediate container 76499115ccf8
--> f4b0016c2dud
Step 4/5 : RUN ["javac", "JavaExample.java"]
--> Running in b8b5434967cc
Removing intermediate container b8b5434967cc
--> 6010eecc9c51
Step 5/5 : ENTRYPOINT ["java","JavaExample"]
--> Running in 04d1092919e0
Removing intermediate container 04d1092919e0
--> 8ccaa003f13e
Successfully built 8ccaa003f13e
Successfully tagged my-java-app:v1 ✓
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions for sensitive files and directories.

```

#docker build -t my-java-example:v1 .

```

C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\8-java-example>docker build -t my-java-example:v1 .
[+] Building 3.5s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 184B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/openjdk:8
=> [internal] load build context
=> => transferring context: 712B
=> CACHED [1/4] FROM docker.io/library/openjdk:8@sha256:bda4739c44b15f4a6ef4aa044e032809dd0ced24263c972bb728a038
=> [2/4] COPY . /src/java
=> [3/4] WORKDIR /src/java
=> [4/4] RUN ["javac", "JavaExample.java"]
=> exporting to image
=> => exporting layers
=> => writing image sha256:3c9c5b6e38933d8258d5f26bea1c0a2a476b272809cd731748665d225ac1c1a
=> => naming to docker.io/library/my-java-example:v1

```

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

```

C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\8-java-example>docker images
REPOSITORY          TAG      IMAGE ID   CREATED        SIZE
my-java-example    v1       3c9c5b6e3893   6 seconds ago  526MB
simple-php-app     latest   f768f160ddb1   5 days ago   452MB
simple-node-server latest   1545cbc9d7d2   5 days ago   172MB

```

Und es funktioniert.

```

C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\8-java-example>docker run 3c9
Hi, Docker demo is here - v2 :)
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\8-java-example>

```

Aufgabe 2 - Erstellen Sie einen Folder auf Rechner und mit diesem Folder erstellen Sie einen neuen Workspace im VS Code. Und danach herstellen die Folders und Dateien wie unten. Schreiben Sie die oben genannte Befehle in [README.md](#) Datei und in die Dockerfile (Wir nennen die Befehl Schritte als LAYERs - LAYER STRUCTURE) für die Automatisierung. Und danach um das Image zu herstellen;

docker build .

```

1-node-app-ubuntu > README.md
1 apt-get update
2 apt-get install curl -y
3 curl -sL https://deb.nodesource.com/setup_10.x | bash
4 apt-get install nodejs -y
5 cd opt
6 mkdir node-app
7 cd node-app/
8 echo 'console.log("nodejsapp from ubuntu ...");' > index.js

```

```

1 FROM ubuntu:18.04
2 RUN apt-get update
3 RUN apt-get install curl -y
4 RUN curl -sL https://deb.nodesource.com/setup_10.x | bash
5 RUN apt-get install nodejs -y
6 COPY . /opt/node-app
7 WORKDIR /opt/node-app
8 #CMD ["node", "/opt/node-app/index.js"];
9 CMD ["node", "index.js"];

```

```

9 | 13 node index.js
10

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
+ v
PS C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\1-node-app-ubuntu> docker build .
[+] Building 0.1s (1/2)

=> => transferring dockerfile: 310B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:18.04
=> [internal] load build context
=> => transferring context: 723B
=> [1/7] FROM docker.io/library/ubuntu:18.04
=> [2/7] RUN apt-get update
=> [3/7] RUN apt-get install curl -y
=> [4/7] RUN curl -sL https://deb.nodesource.com/setup_10.x | bash
=> [5/7] RUN apt-get install nodejs -y
=> [6/7] COPY . /opt/node-app/
=> [7/7] WORKDIR /opt/node-app
=> exporting to image
=> => exporting layers
=> => writing image sha256:19bc13742b2b11a74d1b008ac869e762a1b87f827775502a355be7822cb08904

```

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

```
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\1-node-app-ubuntu>docker images
REPOSITORY          TAG      IMAGE ID   CREATED        SIZE
<none>            <none>   19bc13742b2b  3 minutes ago  240MB
mongo              latest   20106db9aa7a  7 days ago   700MB
ubuntu             latest   3f4714ee068a  7 days ago   77.8MB
ubuntu             18.04   fdf0753c97a9  7 days ago   63.2MB
redis              5        7891e1b96087  8 days ago   110MB
redis              latest   3c3da61c4be0  8 days ago   113MB
mysql              latest   f2ad9f23df82  9 days ago   521MB
docker/getting-started  latest   cb90f98fd791  2 weeks ago  28.8MB
phpmyadmin/phpmyadmin  latest   5682e7556577  2 months ago  524MB
gkandemir/todo-app    latest   ec5c47421d25  14 months ago  181MB
gkandemir/interactive-terminal-app  latest   cc9ff3f96d0b  15 months ago  167MB
gkandemir/counter-app   latest   dc6eb63b9bfc  15 months ago  167MB
gkandemir/node-app     latest   917080a94ca3  15 months ago  238MB

```

Wir können den Name des Images ändern und danach das Image laufen lassen;

```
# docker build . -t simple-node-app und
```

```
# docker run simple-node-app
```

```

=> [6/7] COPY . /opt/node-app/
=> [7/7] WORKDIR /opt/node-app
=> exporting to image
=> => exporting layers
=> => writing image sha256:19bc13742b2b11a74d1b008ac869e762a1b87f827775502a355be7822cb08904
=> => naming to docker.io/library/simple-node-app

```

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

```
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\1-node-app-ubuntu>docker images
REPOSITORY          TAG      IMAGE ID   CREATED        SIZE
simple-node-app    latest   19bc13742b2b  12 minutes ago  240MB
mongo              latest   20106db9aa7a  7 days ago   700MB
ubuntu             latest   3f4714ee068a  7 days ago   77.8MB
ubuntu             18.04   fdf0753c97a9  7 days ago   63.2MB
redis              5        7891e1b96087  8 days ago   110MB
redis              latest   3c3da61c4be0  8 days ago   113MB
mysql              latest   f2ad9f23df82  9 days ago   521MB
docker/getting-started  latest   cb90f98fd791  2 weeks ago  28.8MB
phpmyadmin/phpmyadmin  latest   5682e7556577  2 months ago  524MB
gkandemir/todo-app    latest   ec5c47421d25  14 months ago  181MB
gkandemir/interactive-terminal-app  latest   cc9ff3f96d0b  15 months ago  167MB
gkandemir/counter-app   latest   dc6eb63b9bfc  15 months ago  167MB
gkandemir/node-app     latest   917080a94ca3  15 months ago  238MB

```

```
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\1-node-app-ubuntu>docker run simple-node-app
Hello From Ubuntu 🌟 😊
```

Aufgabe 3 - Jetzt lernen wir eine praktischere und einfachere Methode zum Erstellen eines Images lernen.

In der Webseite können wir ein Version von node wählen und wir können einen fertigen Code verwenden



Quick reference

- Maintained by The Node.js Docker Team
- Where to get help: the Docker Community Forums, the Docker Community Slack, or Stack Overflow

Supported tags and respective Dockerfile links

- 18-alpine3.14, 18.0-alpine3.14, 18.0.0-alpine3.14, alpine3.14, current-alpine3.14
- 18-alpine, 18-alpine3.15, 18.0-alpine, 18.0-alpine3.15, 18.0.0-alpine, 18.0.0-alpine3.15, alpine, alpine3.15, current-alpine, current-alpine3.15
- 18, 18-bullseye, 18.0, 18.0-bullseye, 18.0.0, 18.0.0-bullseye, bullseye, current, current-bullseye, latest
- 18-bullseye-slim, 18-slim, 18.0-bullseye-slim, 18.0-slim, 18.0.0-bullseye-slim, 18.0.0-slim, bullseye-slim, current-bullseye-slim, current-slim, slim
- 18-buster, 18.0-buster, 18.0.0-buster, buster, current-buster
- 18-buster-slim, 18.0-buster-slim, 18.0.0-buster-slim, buster-slim, current-buster-slim

Erstellen Sie einen neuen Folder und Datei im VS Code

```
Dockerfile x
1-node-app-ubuntu > Dockerfile > FROM ubuntu:18.04
1. RUN apt-get update
2. RUN apt-get install curl -y
3. RUN curl -sL https://deb.nodesource.com/setup_10.x | bash
4. RUN apt-get install nodejs -y
5. WORKDIR /opt/node-app
6. COPY . /opt/node-app/
7. ENV channel=kablosuzkedi
8. #CMD ["node", "/opt/node-app/index.js"];
9. CMD ["node", "index.js"];
```



```
Dockerfile x
2-node-app > Dockerfile > ...
1. FROM node:14-slim ✓
2. WORKDIR /opt/node-app
3. COPY . .
4. ENV channel=kablosuzkedi
5. CMD ["node", "index.js"];
```

```
# docker build . -t simple-node-app2
```

```
=> => writing image sha256:6ee020a25a4614a8c0928fcdd5277513ace9d59c89fbe3f601e516c8d4da4a0d
=> => naming to docker.io/library/simple-node-app2
```

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

```
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\2-node-app>docker images
REPOSITORY          TAG      IMAGE ID   CREATED        SIZE
simple-node-app2    latest   6ee020a25a46  About a minute ago  169MB
simple-node-app     latest   18ac4bea72d3  32 minutes ago   240MB
<none>              <none>   63ddd292bc5d  36 minutes ago   240MB
<none>              <none>   055e74414889  46 minutes ago   240MB
<none>              <none>   9ea125bd0e09  49 minutes ago   240MB
```

```
# docker run simple-node-app2
```

```
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\2-node-app>docker run simple-node-app2
Hello From Ubuntu 😊
```

```
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\2-node-app>
```

Aufgabe 4 - Node Server Anwendung

Erstellen Sie app.js Date unter den Aufgabe Folder

```
Dockerfile x
app.js x
package.json
```

```
3-node-server > app.js > app.get("/") callback
1. const express = require("express");
2. const app = express();
3. const PORT = process.env.PORT || 3000;
4.
5. app.get("/", (req, res) => {
6.   res.send("Hello World!");
7.});
```

```

8
9 app.listen(PORT, () =>{
10 |   console.log(`Example app listening at http://localhost:\${PORT}`)
11 });

```

und im neuen Terminal erstellen package.json Date mit dem Befehl;

```
# npm -y init und danach
```

```
# npm install --save express
```

File Edit Selection View Go Run Terminal Help package.json - RESOURCES - Visual Studio Code

Dockerfile x Dockerfile x app.js x package.json x

1-node-app-ubuntu > Dockerfile > FROM

```

1 FROM ubuntu:18.04
2 RUN apt-get update
3 RUN apt-get install curl -y
4 RUN curl -sL https://deb.nodesource.com/setup_10.x | bash
5 RUN apt-get install nodejs -y
6 WORKDIR /opt/node-app
7 COPY . /opt/node-app/
8 ENV channel=kablosuzkedi
9 #CMD ["node", "/opt/node-app/index.js"];
10 CMD ["node", "index.js"];

```

3-node-server > package.json > ...

```

1 {
2   "name": "3-node-server",
3   "version": "1.0.0",
4   "description": "",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \\"Warning: no test specified\\"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC",
12  "dependencies": {
13    "express": "^4.18.0"
14  }
15 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\3-node-server> **npm install --save express**

added 57 packages, and audited 58 packages in 1s

Nun können wir app.js Datei laufen lassen;

```
# node app.js
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

+ v ^ x

pshell pshell pshell pshell pshell

PS C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\3-node-server> **npm install --save express**

added 57 packages, and audited 58 packages in 1s

7 packages are looking for funding
run `npm fund` for details

found 0 vulnerabilities

PS C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\3-node-server> **node app.js**

PS C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\3-node-server> **node app.js**

Example app listening at <http://localhost:3000>

PS C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\3-node-server> []

app.js benötigt zum Ausführen node_modules.

Wir erstellen die Docker-Datei.

.dockerignore x Dockerfile x

3-node-server > .dockerignore > ...

JS app.js x

```

1 node_modules/

```

3-node-server > JS app.js > ...

```

1 const express = require("express");
2 const app = express();
3 const PORT = process.env.PORT || 3000;
4
5 app.get("/", (req, res) => {
6   res.send("Hello World!");
7 });
8
9 app.listen(PORT, () =>{
10   console.log(`Example app listening at http://localhost:\${PORT}`)
11 });

```

Dockerfile x

```

3-node-server > Dockerfile > FROM
1 FROM node:14-slim
2 WORKDIR /opt/node-server
3 COPY . .
4 RUN npm install
5 CMD ["node", "app.js"]

```

C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\2-node-app> docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE

simple-node-server	latest	1545cbc9d7d2	About a minute ago	172MB
simple-node-app2	latest	6ee020a25a46	2 hours ago	169MB
simple-node-app	latest	18ac4bea72d3	2 hours ago	240MB
<none>	<none>	63ddd292bc5d	2 hours ago	240MB
<none>	<none>	055e74414889	2 hours ago	240MB
<none>	<none>	8ea125bd0e99	2 hours ago	240MB

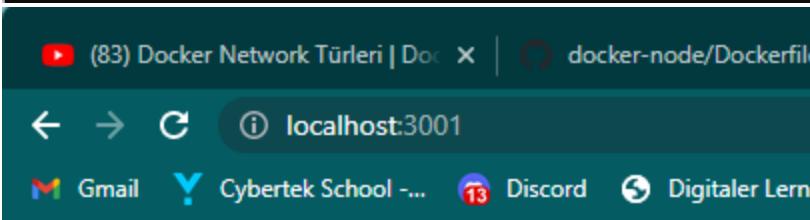
```
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\3-node-server>docker run simple-node-server
Example app listening at http://localhost:3000
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
8f3611343d81	simple-node-server	"docker-entrypoint.s..."	About a minute ago	Up About a minute		funny_sh

```
C:\Users\Selami Demiral>docker stop 8f
8f
C:\Users\Selami Demiral>
```

Jetzt wollen wir Port-Mapping durchführen und auf diese Anwendung erreichen.

```
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\3-node-server>docker run -p 3001:3000 simple-node-server
Example app listening at http://localhost:3000
```



Hello World!

Aufgabe 5 - Jetzt erstellen wir eine Docker-Datei für eine PHP-Anwendung.

```

Dockerfile
FROM php:7-apache
COPY index.php /var/www/html/
EXPOSE 80
CMD ["/usr/sbin/apache2ctl", "-D", "foreground"]

index.php
<?php
echo "Dockerizieren PHP Anwendung ... | kablosuzkedi";
?>

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

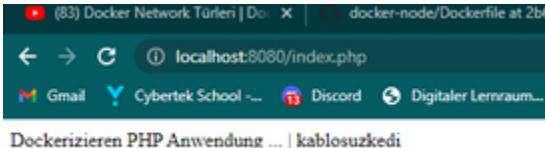
=> extracting sha256:9d02b88c86184fb178b62074c3f76c885c1521cdf98541b49b641cc241004f0
=> extracting sha256:50a246031d43e09937f7cbaba243d807b0a16c917d26c17a7a39e521250c1a1d
=> extracting sha256:a6c0267e6c346f68cc7e514ba9b84716e7cc5542e5be22e1457dec0d5962764
=> extracting sha256:787ca6348cef43213e0cc7654f218c6d91a6ff3e84d26a938bb2926a7cc90c
=> extracting sha256:da8ad43595e2730d6c9054e08ecb6b1357597659a503e77331e4b824db638ac
=> extracting sha256:e191f9e80e295f3e236722e2888e409fd5886667c80fd7488469ae3807e240f
=> naming to docker.io/library/simple-php-app

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
PS C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\4-php-app> 
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
simple-php-app	latest	f768f160ddb1	2 minutes ago	452MB
simple-node-server	latest	1545cbc9d7d2	37 minutes ago	172MB
simple-node-app2	latest	6ee020a25a46	2 hours ago	169MB
simple-node-app	latest	18ac4bea72d3	3 hours ago	240MB
<none>	<none>	63ddd292bc5d	2 hours ago	240MB

```
# docker run -p 8080:80 simple-php-app
```

```
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\3-node-server>docker run -p 8080:80 simple-php-app
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
[Fri Apr 29 13:46:59.146157 2022] [mpm_prefork:notice] [pid 8] AH00163: Apache/2.4.53 (Debian) PHP/7.4.29 configured -- resuming normal operations
[Fri Apr 29 13:46:59.146280 2022] [core:notice] [pid 8] AH00094: Command line: '/usr/sbin/apache2 -D FOREGROUND'
```



7. VERÖFFENTLICHUNG DES NEUEN IMAGES IN DOCKER HUB

Wir wollen jetzt endlich unser Docker Image im Docker Hub ablegen, damit zahlreiche andere Entwickler von unserer Arbeit profitieren können. Dazu sind folgende nötig:

Erstellung eines Docker Hub Repository

The screenshot shows the Docker Hub interface for creating a new repository. The top navigation bar includes 'Explore', 'Repositories', 'Organizations', and 'Help'. A blue header bar has 'demirals' and a dropdown menu. Below the header, it says 'Using 0 of 1 private repositories. [Get more](#)'. The main form has a dropdown 'demirals' and a text input 'docker_lessons' which is highlighted with a yellow box. A 'Description' field is empty. To the right, a 'Pro tip' box contains the command: `docker tag local-image:tagname new-repo:tagname` and `docker push new-repo:tagname`. Below the tip, a note says 'Make sure to change `tagname` with your desired image repository tag.' At the bottom are 'Cancel' and 'Create' buttons.

Melden Sie sich als Nächstes über die Befehlszeile mit dem Docker-Anmeldebefehl bei Docker Hub an.

```
# docker login
```

Der nächste Schritt besteht darin, das Ubuntu-Image zu markieren, damit der Daemon weiß, in welches Repository das Image verschoben werden soll. Dazu können wir den Docker-Tag-Befehl verwenden.

```
# docker tag local-image:tag-name username/reponame:tagname
```

```
# docker push <username>/<reponame>:<tagname>
```

```
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\8-java-example>docker tag docker_lessons:v1 demirals/docker_lessons:v1
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\8-java-example>docker push demirals/docker_lessons:v1
The push refers to repository [docker.io/demirals/docker_lessons]
2a8cb9a09524: Pushed
5f70bf18a086: Pushed
79b1a7ff88ed9: Pushed
575b1387983f: Pushed
0f3078493dc2: Pushed
cac2fff6ae3d: Pushed
08fa02ce37eb: Pushed
a037458de4e0: Pushed
bafdbe68e4ae: Pushed
a13c519c6361: Pushed
v1: digest: sha256:75c8ddb439a50cdf05379d981e4ddee67fb9493cdb22539fa833e0762caaf18 size: 2415
```

The screenshot shows the Docker Hub repository page for 'docker_lessons'. It has a blue header bar with 'demirals' and a dropdown. Below it, it says 'Using 0 of 1 private repositories. [Get more](#)'. The main content area shows a table with one row for 'General'. The 'General' tab is selected. It shows the repository name 'docker_lessons' and a link to 'Advanced Image Management'. A note says 'View preview'.

demirals / docker_lessons

This repository does not have a description

Last pushed: 10 minutes ago

Docker commands

To push a new tag to this repository,

```
docker push demirals/docker_lessons:tagname
```

Tags and Scans

This repository contains 1 tag(s).

TAG	OS	PULLED	PUSHED
v1	✓	---	10 minutes ago

[See all](#)

VULNERABILITY SCANNING - DISABLED

[Enable](#)

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions.

[Upgrade to Pro](#) [Learn more](#)

8. DOCKER COMPOSE

Tutorial Links :

Docker Handbuch für Einsteiger, Hans-M.Hopp

https://www.youtube.com/watch?v=cu3_IdKZ0os&t=1115s

Wir brauchen Docker Compose, um mehrere Dienste harmonisch miteinander auszuführen.

Um die Konfiguration und die Kontrolle komplexer Systeme zu erleichtern, stellt uns Docker das Tool Docker Compose zur Verfügung. Damit wird es möglich, komplexe Lösungen mit mehreren Containern zu definieren, zu verwalten, zu starten und auch wieder anzuhalten.

Die folgenden Merkmale kennzeichnen die Vorteile von Docker Compose:

- Mehrere entkoppelte Laufzeitumgebungen können auf dem gleichen Host-Rechner ausgeführt werden. Es kann über Docker Compose ein Projektname vergeben werden, um die Laufzeitumgebungen zu entkoppeln.
- Erhaltung von Volume-Daten, welche von Services erstellt beziehungsweise benötigt werden. Werden Container erneut gestartet, stehen alle Daten aus vorherigen Container-Versionen immer noch bereit.
- Docker-Compose bietet die inkrementelle Aktualisierung der Container für Services an. Nur Container, welche Änderungen aufweisen, werden vor einem Start neu gebaut. Änderungen an einem Service nehmen dadurch sehr wenig Zeit in Anspruch.
- Compose erlaubt es, Umgebungsvariablen in Compose-Dateien zu benutzen. Mit diesen Variablen kann man beispielsweise eine Komposition an verschiedene Umgebungsbedingungen und Benutzer anpassen.

Docker Compose kann Container nur auf dem lokalen Host starten. Docker Compose eignet sich also nicht für Lösungen zur Hochverfügbarkeit von Diensten und kann auch nicht das Load-Balancing verwalten, also die Lastenverteilung von Diensten. Für diese Aufgaben kommen Orchestrierungs-Anwendungen wie Docker Swarm oder Kubernetes zum Einsatz.

Die Konfiguration von Applikationen erfolgt über eine Konfigurationsdatei, die im YAML-Format erstellt wird. Mit einem einzigen Kommando werden alle benötigten Container-Dienste aus der Konfigurationsdatei heraus gestartet.

Installation von Docker Compose :

Falls Sie Docker Desktop unter Windows oder unter MAC-OS installiert haben, dann beinhaltet diese Installation bereits Docker Compose.

```
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyaları\DOCKER\RESOURCES\8-java-example>docker compose version
Docker Compose version v2.4.1
```

Das YAML-Format :

Docker Compose benutzt eine Textdatei, mit deren Hilfe man die Dienste einer Applikation konfiguriert. Diese Datei muss im YAML-Format erstellt werden und muss unter dem Namen 'dockercompose.yml' oder 'dockercompose.yaml' abgespeichert werden.

Docker-Compose-Dateien spezifizieren die Eigenschaften von Services, Netzwerken und Volumes für Docker Applikationen.

docker run command

```
docker run -d \
--name mongodb \
-p 27017:27017 \
-e MONGO_INITDB_ROOT_USERNAME=admin \
-e MONGO_INITDB_ROOT_PASSWORD=password \
--net mongo-network \
mongo
```

mongo-docker-compose.yaml

```
version: '3'
services:
  mongodb:
    image: mongo
    ports:
      - 27017:27017
    environment:
      - MONGO_INITDB_ROOT_USERNAME=admin
      - MONGO_INITDB_ROOT_PASSWORD=password
```

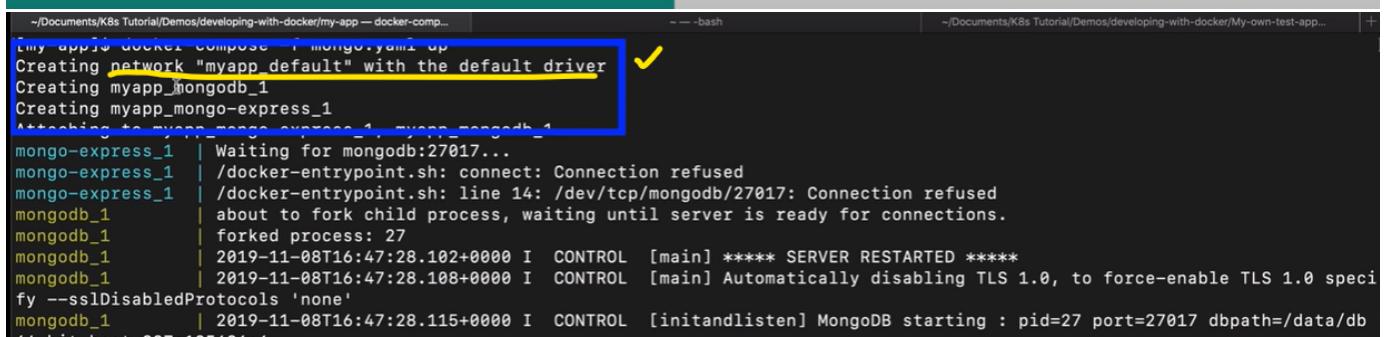
docker run command

```
docker run -d \
--net mongo-network \
...
...
```

mongo-docker-compose.yaml

```
version: '3'
services:
  mongodb:
    image: mongo
    ...
  mongo-express:
    image: mongo-express
    ...
...
```

Docker Compose takes care
of creating a common Network!



```
~/Documents/K8s Tutorial/Demos/developing-with-docker/my-app -- docker-compose up
Creating network "myapp_default" with the default driver
Creating myapp_mongodb_1
Creating myapp_mongo-express_1
Attaching to myapp_mongo-express_1, myapp_mongodb_1
mongo-express_1 | Waiting for mongodb:27017...
mongo-express_1 | /docker-entrypoint.sh: connect: Connection refused
mongo-express_1 | /docker-entrypoint.sh: line 14: /dev/tcp/mongodb/27017: Connection refused
mongodb_1 | about to fork child process, waiting until server is ready for connections.
mongodb_1 | forked process: 27
mongodb_1 | 2019-11-08T16:47:28.102+0000 I CONTROL [main] ***** SERVER RESTARTED *****
mongodb_1 | 2019-11-08T16:47:28.108+0000 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
mongodb_1 | 2019-11-08T16:47:28.115+0000 I CONTROL [initandlisten] MongoDB starting : pid=27 port=27017 dbpath=/data/db 64-bit host=887a135686a6
```

mongo.yam

index.html docker-commands.md mongo.yaml package.json server.js

```
1 version: '3'
2 services:
3   mongodb:
4     image: mongo
5     ports:
```

```

6   - 2017:2701/
7   environment:
8     - MONGO_INITDB_ROOT_USERNAME=admin
9     - MONGO_INITDB_ROOT_PASSWORD=password
10  mongo-express:
11    image: mongo-express
12    ports:
13      - 8080:8081
14    environment:
15      - ME_CONFIG_MONGODB_ADMINUSERNAME=admin
16      - ME_CONFIG_MONGODB_ADMINPASSWORD=password
17      - ME_CONFIG_MONGODB_SERVER=mongodb

```

**Indentation in yaml-File
is important!**

Beispiel-1 mit nur einem Service :

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure with several folders like '3-node-server', '4-php-app', '5-node-mongo-todo', '8-java-example', and '9-docker-compose\1-node-server'. The 'app' folder under '9-docker-compose\1-node-server' is highlighted with a yellow bracket.
- Dockerfile:** The 'Dockerfile' tab shows the following code:

```

1 FROM node:14-slim
2 WORKDIR /opt/node-server
3 COPY app/ .
4 RUN npm install
5 # EXPOSE 3000
6 CMD ["node", "app.js"]

```

A yellow bracket highlights the 'app' directory in the 'COPY' command.
- Info Bar:** A message bar at the bottom right says: "You have Docker installed on your system. Do you want to install the recommended extensions for it?". It has 'Install' and 'Show Recommendations' buttons.

The screenshot shows the 'docker-compose.yml' file with the following content:

```

version: '3.4'
services:
  node-server:
    build: .
    ports:
      - 3001:3000

```

A yellow bracket highlights the port mapping '3001:3000'.

```

# docker-compose build
# docker-compose up
Microsoft Windows [Version 10.0.22000.556]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

```

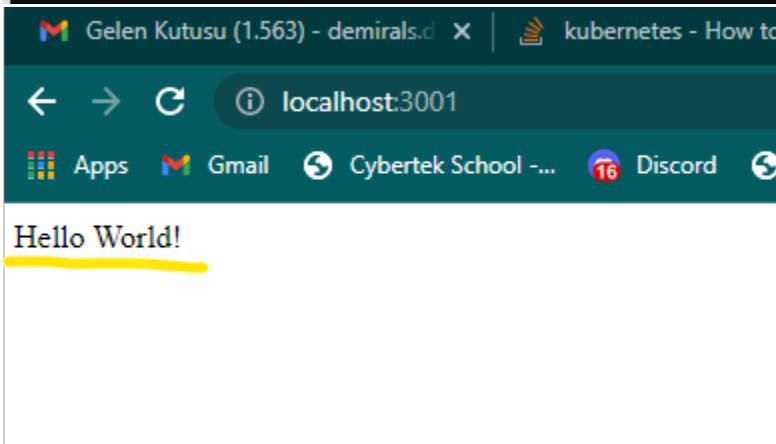
```
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\9-docker-compose\1-node-server>docker-compose build
[+] Building 5.9s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 153B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/node:14-slim
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 38.24kB
=> [1/4] FROM docker.io/library/node:14-slim@sha256:9ea3dfdff723469a060d1fa80577a090e14ed28157334d649518ef7ef8ba5b9b
=> CACHED [2/4] WORKDIR /opt/node-server
=> [3/4] COPY app/ .
=> [4/4] RUN npm install
=> exporting to image
=> => exporting layers
=> => writing image sha256:1ff63b175e92fff7aa64a37b82d0ec294fbcaf368e5f200494dbd95a805a3a6
=> => naming to docker.io/library/1-node-server_node-server

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\9-docker-compose\1-node-server>docker images
REPOSITORY          TAG      IMAGE ID   CREATED        SIZE
1-node-server_node-server  ✓    latest    1ff63b175e92  36 seconds ago  172MB
docker_lessons/java-example  v1      3c9c5b6e3893  6 hours ago   526MB
docker_lessons          v1      3c9c5b6e3893  6 hours ago   526MB
demirals/docker_lessons  v1      3c9c5b6e3893  6 hours ago   526MB
simple-php-app         latest   f768f160ddb1  6 days ago   452MB
simple-node-server     latest   1545cbc9d7d2  6 days ago   172MB
simple-node-app2       latest   6ee020a25a46  6 days ago   169MB
simple-node-app        latest   18ac4bea72d3  6 days ago   240MB
```

```
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\9-docker-compose\1-node-server>docker-compose up ✓
[+] Running 1/1
- Container 1-node-server-node-server-1 Recreated
Attaching to 1-node-server-node-server-1
1-node-server-node-server-1 | Example app listening at http://localhost:3000 😊
```



Um den bestehenden Container umbenennen :

```
9-docker-compose > 1-node-server > ↵ docker-compose.yml
 1  version: '3.4'
 2  services:
 3    node-server:
 4      container_name: my-server ✓
 5      build: .
 6      ports:
 7        - 3001:3000
```

```
- Container 1-node-server-node-server-1 Stopped ✓
canceled
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\9-docker-compose\1-node-server>docker-compose build
```

```
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\9-docker-compose\1-node-server>docker-compose build
[+] Building 1.7s (10/10) FINISHED
-> [internal] load build definition from Dockerfile
-> => transferring dockerfile: 31B
-> [internal] load .dockerignore
-> => transferring context: 2B
-> [internal] load metadata for docker.io/library/node:14-slim
-> [auth] library/node:pull token for registry-1.docker.io
-> [1/4] FROM docker.io/library/node:14-slim@sha256:9ea3dfdff723469a060d1fa80577a090e14ed28157334d649518ef7ef8ba5b9b
-> [internal] load build context
-> => transferring context: 133B
-> CACHED [2/4] WORKDIR /opt/node-server
-> CACHED [3/4] COPY app/ .
-> CACHED [4/4] RUN npm install
-> exporting to image
-> => exporting layers
-> => writing image sha256:8b677be812026d9a8a4c5decb19e053429a87dfd4e31136c0a9cd95ef8672edc
-> => naming to docker.io/library/1-node-server_node-server

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\9-docker-compose\1-node-server>docker-compose up
[+] Running 1/1
 - Container 1-node-server-node-server-1 Recreated
Attaching to my-server
my-server | Example app listening at http://localhost:3000
```

Beispiel-2 mit mehreren Services :

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows a tree view of project files. A yellow bracket highlights the folder structure under '9-docker-compose > 2-todo-app'. Inside '2-todo-app', there is a 'app' folder containing 'app.js', 'package-lock.json', 'package.json', and 'TodoModel.js'. Below 'app' is another 'docker-compose.yml' file.
- Code Editor (Right):** Displays the contents of the 'docker-compose.yml' file for the 'todo-app' service. The file defines a container named 'dc-todo-app' built from the current directory, port 3000 mapped to 3000, and uses a 'mongodb' image with port 27017 mapped. It also defines a volume 'todo-app-data' mounted at '/data/db'.
- Handwritten Note:** A yellow checkmark is placed next to the 'mongodb' section, with the handwritten text '✓ vom hub' written next to it.

```
version: '3.4'
services:
  todo-app:
    container_name: dc-todo-app
    build: .
    ports:
      - 3000:3000
  mongodb:
    image: mongo
    ports:
      - 27017:27017
    volumes:
      - todo-app-data:/data/db
volumes:
  todo-app-data:
```

Link für die Dateien <https://github.com/gkandemi/docker/tree/main/apps/9-docker-compose/2-todo-app>

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the 'Dockerfile' file selected in the '2-todo-app' folder.
- Code Editor (Right):** Displays the Dockerfile content for the 'todo-app' service. It specifies the base image as 'node:14-slim', sets the working directory to '/opt/node-todo-app', copies the 'app' directory to the working directory, runs 'npm install', and sets the command to 'node app.js'.

```
FROM node:14-slim
WORKDIR /opt/node-todo-app
COPY app/ .
RUN npm install
CMD ["node", "app.js"]
```

```
C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\9-docker-compose\2-todo-app>docker-compose build
```

```
[+] Building 7.4s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 136B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/node:14-slim
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 1.26kB
=> [1/4] FROM docker.io/library/node:14-slim@sha256:9ea3dfdff723469a060d1fa80577a090e14ed28157334d649518ef7ef8ba5b9b
=> CACHED [2/4] WORKDIR /opt/node-todo-app
=> [3/4] COPY app/
=> [4/4] RUN npm install
=> exporting to image
=> => exporting layers
=> => writing image sha256:6279f944fe6a6800756d3d135beefde39d4364df2f8171891265f47b06e0bc9b
=> => naming to docker.io/library/2-todo-app_todo-app

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyalari\DOCKER\RESOURCES\9-docker-compose\2-todo-app>docker-compose up
[+] Running 2/2
 - Container 2-todo-app-mongodb-1  Created
 - Container dc-todo-app  Recreated
metadata', 'attr': {'remote': '172.19.0.2:45420', 'client': 'conn1', 'doc': {'driver': {'name': 'nodejs|Mongoose', 'version': '3.6.4'}, 'os': {'type': 'Linux', 'name': 'linux', 'architecture': 'x64', 'version': '5.10.16.3-microsoft-standard-WSL2'}, 'platform': "'Node.js v14.19.1, LE (unified)", "version": "3.6.4[5.11.16]'}}}
dc-todo-app | (node:1) DeprecationWarning: Listening to events on the Db class has been deprecated and will be removed in the next major version.
dc-todo-app | MongoDB'ye bağlantı başarılı! 😊
2-todo-app-mongodb-1 | {"t": {"$date": "2022-05-05T15:21:16.754+00:00"}, "s": "I", "c": "NETWORK", "id": 22943, "ctx": "listener", "msg": "Connection accepted", "attr": {"remote": "172.19.0.2:45422", "uuid": "2f91055c-cac6-4d8f-805c-cad11766baa6", "connectionId": 2, "connectionCount": 2}}
C:\Users\Selami Demiral>docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
73cd7121e439 2-todo-app_todo-app "docker-entrypoint.s..." 15 seconds ago Up 12 seconds 0.0.0.0:3000->3000/tcp dc-todo-app
9fa01e87c69b mongo "docker-entrypoint.s..." 15 minutes ago Up 12 seconds 0.0.0.0:27017->27017/tcp 2-todo-app-mongodb-1
C:\Users\Selami Demiral>
```

New Collection / New Request

POST http://localhost:3000/todo

Params Auth Headers (8) Body **JSON** Pre-req. Tests Settings

```
1
2   ...
3   ...
4   ...
5   ...
6   ...
7   ...
8   ...
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1
2   ...
3   ...
4   ...
5   ...
6   ...
7   ...
8   ...
```

200 OK 10 ms 414 B Save Response

New Collection / New Request

GET http://localhost:3000

Params Auth Headers (8) Body **JSON** Pre-req. Tests Settings

Save Cookies

Beautify

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

```

C:\Users\Selami Demiral\Desktop\dokumanlar\PROGRAM exe Dosyaları\DOCKER\RESOURCES\9-docker-compose\2-todo-app>docker-compose down

[+] Running 3/3

- Container 2-todo-app-mongodb-1 Removed
- Container dc-todo-app Removed
- Network 2-todo-app_default Removed

Beispiel-3 Docker Compose mit Python und php :

<https://www.youtube.com/watch?v=Qw9zIE3t8Ko>

- Erstellen Python Datei

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

```

- Docker-Compose.yml Datei

```
1 version: '3'
2
3 services:
4   product-service:
5     build: ./product
6     volumes:
7       - ./product:/usr/src/app
8     ports:
9       - 5001:80
10
11   website:
12     image: php:apache
13     volumes:
14       - ./website:/var/www/html
15     ports:
16       - 5000:80
17     depends_on:
18       - product-service
```

- Mit dem Befehl docker-compose können wir sehen, dass wir beide Container gleichzeitig arbeiten können

```
jake:website jake$ docker-compose up -d
Starting tutorial_product-service_1
Starting tutorial_website_1
jake:website jake$ ls
index.php
jake:website jake$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CF
bdbce876ee35      php:apache          "docker-php-entryp..."    7
ad045ea05f39       tutorial_product-service   "python api.py"        7
jake:website jake$
```

Beispiel-4 Erstellen von Apps mit mehreren Containern mit MySQL und Docker Compose

<https://docs.microsoft.com/de-de/visualstudio/docker/tutorials/tutorial-multi-container-app-mysql>

9. ZUSÄLTICHE INFORMATIONEN

Aktualisieren Ubuntu

```
C:\Users\USER>docker run -it ubuntu:18.04
root@3631a5de17f9:/# apt-get update
Get:1 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:2 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [186 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic/restricted amd64 Packages [13.5 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic/main amd64 Packages [1344 kB]
Get:8 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [909 kB]
Get:9 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [11.3 MB]
Get:10 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [2732 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [2272 kB]
```

```
Get:12 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [1496 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [3167 kB]
Get:14 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [21.1 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [29.8 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [942 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic-backports/main amd64 Packages [12.2 kB]
Get:18 http://archive.ubuntu.com/ubuntu bionic-backports/universe amd64 Packages [12.9 kB]
Fetched 25.0 MB in 9s (2635 kB/s)
Reading package lists... Done
root@3631a5de17f9:/#
```

Installieren Node-Js

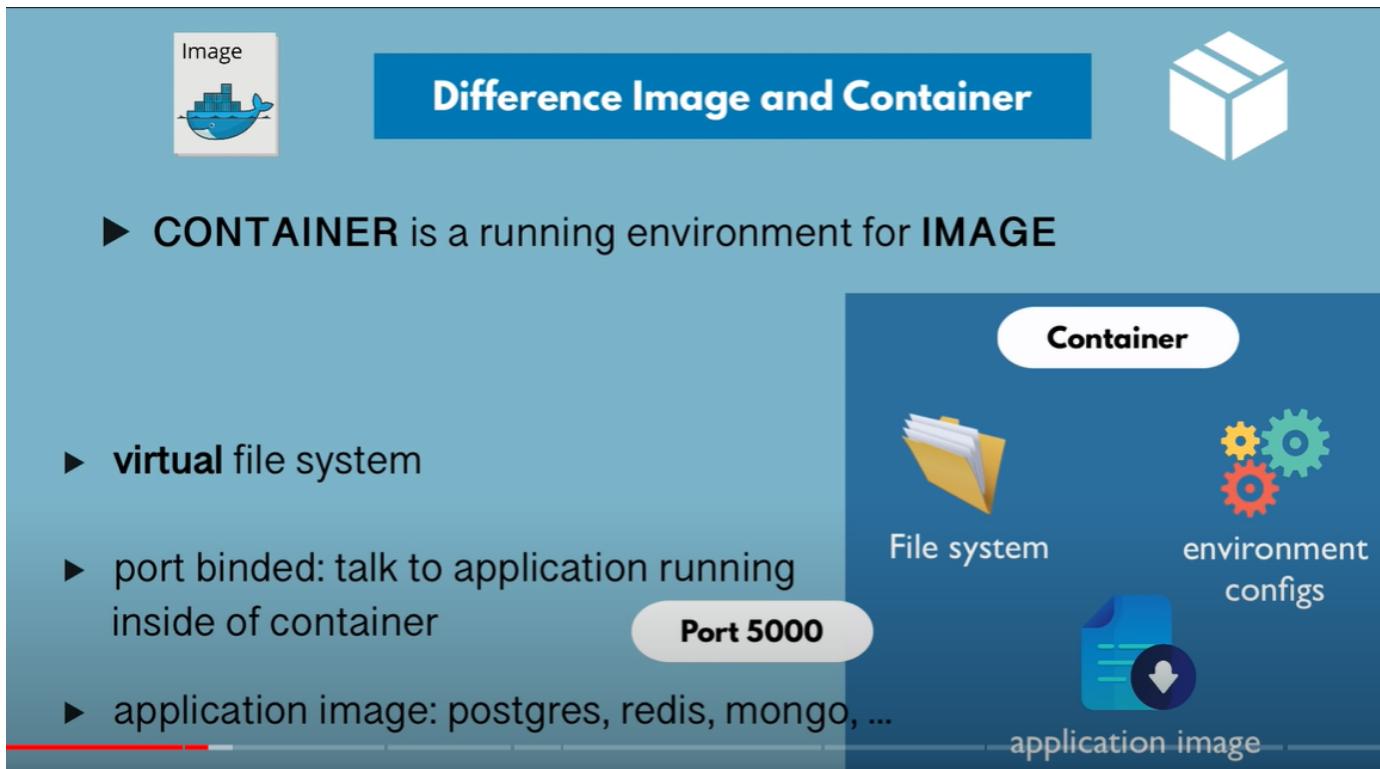
```
C:\Users\USER>curl sL https://deb.nodesource.com/setup_10.x | bash
```

npm init -y. generiert einfach ein leeres npm-Projekt, ohne einen interaktiven Prozess zu durchlaufen. Das steht für ja.

Container vs. Image

Unter Verwendung einer objektorientierten Programmieranalogie ist der Unterschied zwischen einem Docker-Image und einem Docker-Container derselbe wie der Unterschied zwischen einer Klasse und einem Objekt. Ein Objekt ist die Laufzeitinstanz einer Klasse. Ebenso ist ein Container die Laufzeitinstanz eines Images.

Ein Objekt wird nur einmal erstellt, wenn es instanziert wird. Ebenso kann ein Container ausgeführt oder angehalten werden.



Dies sind einige Gründe für die Verwendung von Apps mit mehreren Containern:

- Trennen Sie Container, um APIs und Front-Ends anders zu verwalten als Datenbanken.
- Container ermöglichen das Verwalten und Aktualisieren von Versionen in Isolation.

- Obwohl Sie möglicherweise einen Container lokal für die Datenbank verwenden, sollten Sie in der Produktion einen verwalteten Dienst für die Datenbank nutzen.
- Zum Ausführen mehrerer Prozesse ist ein Prozess-Manager erforderlich, durch den das Starten/Herunterfahren von Containern komplexer wird.

Der Befehl, der erforderlich ist, um den Inhalt eines Images zu erfahren:

```
root@4d0763784aae:/etc# cat os-release
PRETTY_NAME="Ubuntu 22.04 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04 (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
root@4d0763784aae:/etc#
```