Docker Container Cheatsheet

Zusammenfassung der Wichtigsten Commands

- **Neustarten eines gestoppten Containers:**

```
docker container start <Container-ID oder Name>
```

- **Neustarten eines laufenden Containers:**

```
docker container restart <Container-ID oder Name>
```

- **Erstellen eines Containers:**

```
docker container create --publish <Port-Host>:<Port-Container> <Image-Name>
```

- **Entfernen eines gestoppten Containers:**

```
docker container rm <Container-ID oder Name>
```

- **Alle nicht benutzten Container entfernen:**

```
docker container prune
```

- **Laufenden Container mit --rm Option starten (wird nach Stop automatisch entfernt):**

```
docker container run --rm --detach --publish <Port-Host>:<Port-Container> --
name <Container-Name> <Image-Name>
```

Container Ausführen:

- **Alte Syntax (vor Docker 1.13):**

```
docker run <Image-Name>
```

- **Neue Syntax (ab Docker 1.13):**

```
docker container run <Image-Name>
```

Ports Veröffentlichen:

- **Um Ports freizugeben:**

```
docker container run --publish <Host-Port>:<Container-Port> <Image-Name>
```

Im Hintergrund Ausführen (Detached Mode):

- **Container im Hintergrund starten:**

```
docker container run --detach --publish <Host-Port>:<Container-Port> <Image-
Name>
```

Container Auflisten:

- **Laufende Container anzeigen:**

```
docker container ls
```

- **Alle Container anzeigen (auch gestoppte):**

```
docker container ls --all
```

Container Benennen oder Umbenennen:

- **Container mit einem Namen starten:**

```
docker container run --detach --publish <Host-Port>:<Container-Port> --name
<Name> <Image-Name>
```

- **Container umbenennen:**

```
docker container rename <Alter-Container-Name> <Neuer-Name>
```

Container Stoppen oder Beenden:

- **Einen laufenden Container stoppen:**

```
docker container stop <Container-Name-oder-ID>
```

- **Einen laufenden Container sofort beenden (SIGKILL):**

```
docker container kill <Container-Name-oder-ID>
```

Anmerkungen:

- Die Option `--publish` oder `-p` wird verwendet, um Container-Ports auf Host-Ports abzubilden.
- Der Befehl `docker container ls` zeigt nur laufende Container an, es sei denn, die Option `--all` oder `-a` wird verwendet, um alle Container anzuzeigen.
- `--detach` oder `-d` wird verwendet, um Container im Hintergrund auszuführen.
- `--name` gibt einem Container einen spezifischen Namen, damit du ihn leichter identifizieren kannst.
- `docker container stop` sendet zuerst ein SIGTERM und wartet eine Gnadenfrist, bevor ein SIGKILL gesendet wird, falls der Container nicht rechtzeitig stoppt.
- `docker container kill` sendet sofort ein SIGKILL, um den Container zu beenden.

Container Neustarten

- **Bereits gestoppte oder beendete Container neustarten:**

```
docker container start <Container-ID oder Name>
```

- **Liste aller Container anzeigen (auch beendete):**

```
docker container ls --all
```

- **Spezifischen Container neu starten:**

```
docker container restart <Container-ID oder Name>
```

Container Erstellen Ohne Starten

- **Container erstellen:**

```
docker container create --publish 8080:80 <Image-Name>
```

- **Erstellten Container starten:**

```
docker container start <Container-ID oder Name>
```

Nicht Benutzte Container Entfernen

- **Einen bestimmten gestoppten Container entfernen:**

```
docker container rm <Container-ID oder Name>
```

- **Alle nicht benutzten Container auf einmal entfernen:**

```
docker container prune
```

    - Bestätigung umgehen:

```
docker container prune -f
```

Container Mit Automatischer Löschung

- **Container erstellen und automatisch löschen nach Beendigung:**

```
docker container run --rm --detach --publish 8888:80 --name <Container-Name>
<Image-Name>
```

Container Stoppen und Automatisch Entfernen

- **Container stoppen:**

```
docker container stop <Container-Name>
```

- Hier wird der Container automatisch entfernt, falls `--rm` beim Start benutzt wurde.

Interaktive Container ausführen

- **Bash in Ubuntu Container starten**:

```
docker container run --rm -it ubuntu
```

`-it` ermöglicht interaktive Prozesse (`-i` für interaktive Eingaben, `-t` für ein pseudo-TTY).

Befehle innerhalb eines Containers ausführen

- **Befehl in Alpine Container ausführen**:

```
docker run alpine uname -a
```

Führt den `uname -a` Befehl in einem ephemeren Alpine-Container aus.

Mit ausführbaren Images arbeiten

- **rmbyext Script ausführen**:

```
docker container run --rm -v $(pwd):/zone fhsinchy/rmbyext pdf
```

Führt das rmbyext-Programm im Container aus, um PDF-Dateien im aktuellen Verzeichnis zu löschen.

Optionen und Parameter

- `-it`: Kombination aus `-i` und `-t`, notwendig für interaktive Sitzungen.
- `--rm`: Entfernt den Container automatisch nach Ausführung.
- `-v` / `--volume`: Bindet ein lokales Verzeichnis in den Container ein.

Volumes und Bind Mounts

- **Bind Mount Syntax**:

```
--volume <lokales Verzeichnis>:<Verzeichnis im Container>:<Zugriffsrechte>
```

Bindet das lokale Verzeichnis `<lokales Verzeichnis>` an das Verzeichnis `<Verzeichnis im Container>` im Container. `<Zugriffsrechte>` ist optional und steuert die Schreib-/Leseberechtigungen.

Nützliche Befehle und Beispiele

- **Interaktiver Modus**:

```
docker container run -it node
```

  Startet einen Node.js-Container für interaktive JavaScript-Ausführungen.

- **Befehl in einem nicht laufenden Container ausführen**:

```
docker container run <Image-Name> <Befehl>
```

  Führt `<Befehl>` in einem Container basierend auf `<Image-Name>` aus.

- **Base64-Encoding eines Strings**:

```
docker container run --rm busybox sh -c "echo -n my-secret | base64"
```

  Nutzt das BusyBox-Image, um den String "my-secret" zu Base64 zu kodieren.

Hinweise

- **Befehlsübergabe**: Nach dem Image-Namen übergebene Befehle werden an den Entry-Point des Images weitergegeben.
- **Ausführbare Images**: Verhalten sich wie ausführbare Programme und nehmen Argumente für die Ausführung entgegen.