Docker

Docker ist eine freie Software zur Isolierung von Anwendungen mit Hilfe von Containervirtualisierung.

Docker vereinfacht die Bereitstellung von Anwendungen, weil sich Container, die alle nötigen Pakete enthalten, leicht als Dateien transportieren und installieren lassen. Container gewährleisten die Trennung und Verwaltung der auf einem Rechner genutzten Ressourcen. Das umfasst laut Aussage der Entwickler: Code, Laufzeitmodul, Systemwerkzeuge, Systembibliotheken – alles was auf einem Rechner installiert werden kann

Installation

Installieren via APT Paketmanager

Beschreibung:

Installations via APT Paketmanager wie Debian/Ubuntu.

Vorrausetzungen

Paketliste Aktualiseren

```
sudo apt-get update
```

Pakete installieren

```
sudo apt update
sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
```

Docker installieren

```
sudo apt install docker.io docker-compose
```

Docker version anzeigen und Funktionsprüfung

```
docker --version
```

Ausgabe:

```
Docker version 20.10.21, build 20.10.21-0ubuntu1~22.04.2
```

Fertig

Troubleshooting

Troubleshooting

Docker belegt verdammt viel Speicher

Beschreibung:

Auf einmal belegt Docker im /var/lib/docker verdammt viel Speicher.
Im overlay2 Verzeichnis wird folgendes gespeichert:

Das Docker Overlay-Verzeichnis ist Teil des OverlayFS (ein Union-Dateisystem), das Docker für seine Images und Container verwendet. Dieses Dateisystem ermöglicht es, dass mehrere Images Schichten teilen, um Speicherplatz zu sparen, und gleichzeitig die Änderungen, die in Containern gemacht werden, zu isolieren. Hier sind die Hauptkomponenten und Funktionen des Docker Overlay-Verzeichnisses:

1. **Image Layers**: Jedes Docker-Image besteht aus mehreren Schichten (layers), die durch das OverlayFS zusammengefügt werden. Diese Schichten sind nur lesbare Dateien und Verzeichnisse.
2. **Writable Container Layer**: Wenn ein Container gestartet wird, erstellt Docker eine neue schreibbare Schicht oben auf den Image-Schichten. Alle Änderungen, die im Container gemacht werden, etwa das Hinzufügen, Löschen oder Ändern von Dateien, werden in dieser schreibbaren Schicht gespeichert.
3. **Volume Data**: Volumes, die für die dauerhafte Speicherung von Daten verwendet werden, können ebenfalls im Overlay-Verzeichnis gespeichert werden, abhängig von der Konfiguration.

Wenn das Docker Overlay-Verzeichnis unerwartet groß wird, kann das mehrere Ursachen haben:

- **Alte Container und Images**: Nicht mehr verwendete Container und Images können immer noch Speicherplatz belegen. Sie können alte Container und Images mit Befehlen wie `docker container prune`, `docker image prune` oder `docker system prune` aufräumen.
- **Log-Dateien**: Manchmal können Container-Log-Dateien viel Platz einnehmen. Diese befinden sich normalerweise im Verzeichnis `/var/lib/docker/containers/<container-id>/` und können sicher gelöscht werden, wenn sie nicht mehr benötigt werden.
- **Volumes und Netzwerke**: Nicht gelöschte Volumes und Netzwerke können ebenfalls Speicherplatz beanspruchen.

- **Dangling Layers**: Gelegentlich bleiben bei der Erstellung und Löschung von Containern und Images "herrenlose" Schichten zurück, die nicht mehr verwendet werden. Diese können mit `docker system prune` entfernt werden.

Es ist wichtig, regelmäßige Wartungen durchzuführen und nicht benötigte Container, Images, Volumes und Netzwerke zu entfernen, um sicherzustellen, dass das Overlay-Verzeichnis nicht unnötig groß wird.

Kürzübersicht der Befehle

```
alte nicht laufende container aufräumen
docker container prune

alte image wegräumen die nicht benötigt werden
docker image prune

will mann alles auf einmal durchführen
docker system prune
```

Troubleshooting

API Error beim Ausführen

Beschreibung:

beim Ausführen von Docker Compose

```
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/urllib3/connectionpool.py", line 704, in urlopen
    httplib_response = self._make_request(
                       ^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/urllib3/connectionpool.py", line 399, in _make_request
    conn.request(method, url, **httplib_request_kw)
  File "/usr/lib/python3.11/http/client.py", line 1282, in request
    self._send_request(method, url, body, headers, encode_chunked)
  File "/usr/lib/python3.11/http/client.py", line 1328, in _send_request
    self.endheaders(body, encode_chunked=encode_chunked)
  File "/usr/lib/python3.11/http/client.py", line 1277, in endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
  File "/usr/lib/python3.11/http/client.py", line 1037, in _send_output
    self.send(msg)
  File "/usr/lib/python3.11/http/client.py", line 975, in send
    self.connect()
  File "/usr/lib/python3/dist-packages/docker/transport/unixconn.py", line 30, in connect
    sock.connect(self.unix_socket)
FileNotFoundError: [Errno 2] No such file or directory

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/requests/adapters.py", line 489, in send
    resp = conn.urlopen(
           ^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/urllib3/connectionpool.py", line 788, in urlopen
    retries = retries.increment(
              ^^^^^^^^^^^^^^^^^^
```

```
  File "/usr/lib/python3/dist-packages/urllib3/util/retry.py", line 550, in increment
    raise six.reraise(type(error), error, _stacktrace)
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/six.py", line 718, in reraise
    raise value.with_traceback(tb)
  File "/usr/lib/python3/dist-packages/urllib3/connectionpool.py", line 704, in urlopen
    httplib_response = self._make_request(
                       ^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/urllib3/connectionpool.py", line 399, in _make_request
    conn.request(method, url, **httplib_request_kw)
  File "/usr/lib/python3.11/http/client.py", line 1282, in request
    self._send_request(method, url, body, headers, encode_chunked)
  File "/usr/lib/python3.11/http/client.py", line 1328, in _send_request
    self.endheaders(body, encode_chunked=encode_chunked)
  File "/usr/lib/python3.11/http/client.py", line 1277, in endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
  File "/usr/lib/python3.11/http/client.py", line 1037, in _send_output
    self.send(msg)
  File "/usr/lib/python3.11/http/client.py", line 975, in send
    self.connect()
  File "/usr/lib/python3/dist-packages/docker/transport/unixconn.py", line 30, in connect
    sock.connect(self.unix_socket)
urllib3.exceptions.ProtocolError: ('Connection aborted.', FileNotFoundError(2, 'No such file or directory'))

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/docker/api/client.py", line 214, in _retrieve_server_version
    return self.version(api_version=False)["ApiVersion"]
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/docker/api/daemon.py", line 181, in version
    return self._result(self._get(url), json=True)
                        ^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/docker/utils/decorators.py", line 46, in inner
    return f(self, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/docker/api/client.py", line 237, in _get
    return self.get(url, **self._set_request_timeout(kwargs))
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
  File "/usr/lib/python3/dist-packages/requests/sessions.py", line 600, in get
    return self.request("GET", url, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/requests/sessions.py", line 587, in request
    resp = self.send(prep, **send_kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/requests/sessions.py", line 701, in send
    r = adapter.send(request, **kwargs)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/requests/adapters.py", line 547, in send
    raise ConnectionError(err, request=request)
requests.exceptions.ConnectionError: ('Connection aborted.', FileNotFoundError(2, 'No such file or directory'))

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/bin/docker-compose", line 33, in <module>
    sys.exit(load_entry_point('docker-compose==1.29.2', 'console_scripts', 'docker-compose')())

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/compose/cli/main.py", line 81, in main
    command_func()
  File "/usr/lib/python3/dist-packages/compose/cli/main.py", line 200, in perform_command
    project = project_from_options('.', options)
              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/compose/cli/command.py", line 60, in project_from_options
    return get_project(
           ^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/compose/cli/command.py", line 152, in get_project
    client = get_client(
             ^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/compose/cli/docker_client.py", line 41, in get_client
    client = docker_client(
             ^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/compose/cli/docker_client.py", line 170, in docker_client
    client = APIClient(use_ssh_client=not use_paramiko_ssh, **kwargs)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/docker/api/client.py", line 197, in __init__
```

```
    self._version = self._retrieve_server_version()
                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/docker/api/client.py", line 221, in _retrieve_server_version
    raise DockerException(
docker.errors.DockerException: Error while fetching server API version: ('Connection aborted.',
FileNotFoundError(2, 'No such file or directory'))
```

Fix:

Es sieht danach aus als wenn der Docker Dienst nicht läuft / bzw Docker nicht installiert ist oder die URL Lib nicht installiert ist.
Führen wir einfach alles in einem Schritt nochmal durch.
in diesem Schritt können wir auch gleich pip mit installieren falls man es mal braucht ;-)

```
apt install python3-pip python3-urllib3 docker.io
```

AppArmor Error

Beschreibung:

Bei Hetzner zum beispiel oder anderen Systemen kann Apparmor enbaled sein.
Dazu müssen die Apparmor Pakete installiert werden sonst bekommt beim Container starten diesen Fehler:

```
Error response from daemon: AppArmor enabled on system but the docker-default profile could not be loaded:
running `apparmor_parser apparmor_parser --version` failed with output:
error: exec: "apparmor_parser": executable file not found in $PATH
```

Fix:

```
# AppArmor installieren, falls es nicht installiert ist
apt-get install apparmor apparmor-utils
```

Befehle

Befehle

Docker bash / shell einloggen

Beschreibung:

Manchmal möchte man auch gerne direkt im container was nachschauen.
Hier der Befehle um sich in einen docker container ein zu loggen.

Befehle:

Erstmal wissen wie der container heißt.
Entweder die container ID oder den Namen nehhmen. Ich bevorzuge Namem.

```
docker ps
```

Ausgabe

```
root@unvr:~# docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED      STATUS       PORTS     NAMES
375668882f07   markdegroot/unifi-protect-arm64   "/lib/systemd/systemd"   3 days ago   Up 16 hours
unifi-protect
root@unvr:~#



ENtweder container ID: 375668882f07
oder Name:  unifi-protect
```

Nun kann entweder bash oder shell genutzt werden, je nachdem was der Container beinhaltet

```
mit bash
docker exec -it <container-id> bash


mit shell
```

docker exec -it <container-id> sh

Beispiel ID

mit bash
docker exec -it 375668882f07 bash

mit shell
docker exec -it 375668882f07 sh

Beispiel Name

mit bash
docker exec -it unifi-protect bash

mit shell
docker exec -it unifi-protect sh

Befehle

Alle Docker instanzen stoppen und/oder löschen

Lösung:

Alle laufenden Container sofort stoppen

```
docker stop $(docker ps -aq)
```

Alle Container löschen

```
docker rm $(docker ps -a -q)
```

Image(s) löschen

```
docker rmi $(docker images -q)
```

Erklärung der docker ps Parameter:

- `docker ps -q`: Listet die IDs aller laufenden Container auf.
- `docker stop $(docker ps -q)`: Stoppt alle laufenden Container.
- `docker ps -a -q`: Listet die IDs aller Container auf (laufende und gestoppte).
- `docker rm $(docker ps -a -q)`: Löscht alle Container.