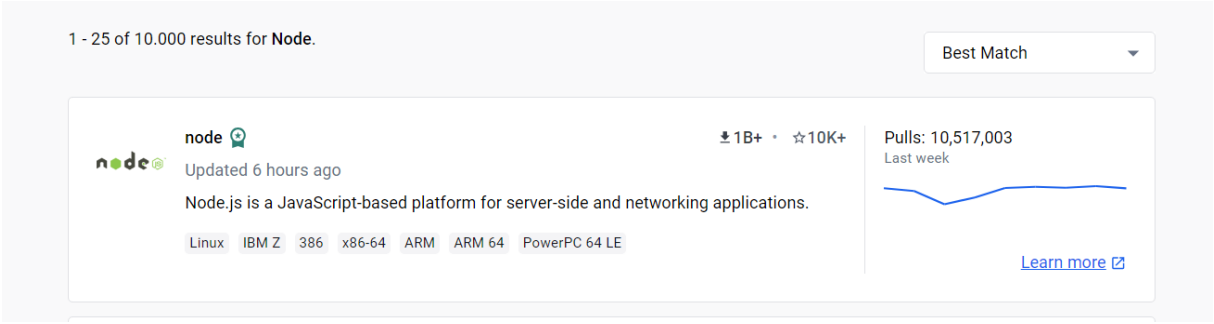
Protokoll zu Docker-Aufgaben

1.) Finde auf Docker Hub das offizielle Node.js Image



2.) Welche drei Hauptvarianten werden in der Dokumentation für das Image aufgeführt und was sind die Unterschiede zwischen ihnen?

a. **node:<version>**: Dies ist das Standard-Node.js-Image, das auf Debian basiert.
b. **node:<version>-alpine**: Dies ist die Alpine-basierte Variante des Node.js-Images.
c. **node:<version>-slim**: Diese Variante basiert auf Debian und ist im Vergleich zur Standardvariante etwas schlanker.

3.) Lade das aktuelle **node:alpine-Image** herunter

```
PS C:\Users\bsulj> docker pull node:alpine
alpine: Pulling from library/node
4abcf2066143: Pull complete
f16884bf7ba3: Downloading [==============>                                    ]   11.5MB/43.52MB
12fe8815c466: Download complete
9b5c6033110b: Download complete
```

4.) Wie groß ist es im Vergleich zum Standard-Node.js-Image?

Die alpine Version ist kleiner als die Standardversion, weil da eine schlankere Linux-Distribution

5.) Benenne das heruntergeladene Image in small-node um

```
PS C:\Users\bsulj> docker tag node:alpine small-node
PS C:\Users\bsulj>
```

6.) Überzeuge dich, dass bei dir jetzt ein Image namens small-node existiert

```
PS C:\Users\bsulj> docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
node         alpine    a4953861f11d   2 weeks ago   141MB
small-node   latest    a4953861f11d   2 weeks ago   141MB
library      latest    a524f0e23179   4 weeks ago   1.42GB
ubuntu       latest    e34e831650c1   4 weeks ago   77.9MB
PS C:\Users\bsulj>
```

7.) Erzeuge und starte einen Node.js-Container basierend auf dem small-node-Image, welcher automatisch wieder gelöscht werden soll. Starte dann per docker exec in diesem Container eine Bash

```
PS C:\Users\bsulj> docker run --rm -it small-node bash
node:internal/modules/cjs/loader:1152
  throw err;
  ^

Error: Cannot find module '/bash'
    at Module._resolveFilename (node:internal/modules/cjs/loader:1149:15)
    at Module._load (node:internal/modules/cjs/loader:990:27)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:142:12)
    at node:internal/main/run_main_module:28:49 {
  code: 'MODULE_NOT_FOUND',
  requireStack: []
}

Node.js v21.6.1
PS C:\Users\bsulj>
```

Bash kann nicht gestartet werden da, der Container gleich wieder gelöscht wird

8.) Welche Fehlermeldung tritt dabei auf?

```
node:internal/modules/cjs/loader:1152
  throw err;
  ^

Error: Cannot find module '/bash'
    at Module._resolveFilename (node:internal/modules/cjs/loader:1149:15)
    at Module._load (node:internal/modules/cjs/loader:990:27)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:142:12)
    at node:internal/main/run_main_module:28:49 {
  code: 'MODULE_NOT_FOUND',
  requireStack: []
}

Node.js v21.6.1
```

9.) Lösche beide Node.js-Images, die auf Alpine basieren

```
PS C:\Users\bsulj> docker rmi node:alpine small-node
Untagged: node:alpine
Untagged: node@sha256:4cc2d9f365691fc6f8fe227321d32d9a2691216a71f51c21c7f02224515dea48
Untagged: small-node:latest
Deleted: sha256:a4953861f11d2c59d7d524dbcc90827fc27bde5ed4ffee7df711319523c8b35d
Deleted: sha256:fa80e2f1dd94196ba9b6b8ad5a54ec213ae1b9aea47d254e78ec67ef87e4c564
Deleted: sha256:2e3faf178587fe5d1efb0c82349b6372fdcb8aaf653d118bf97479a5a43cf631
Deleted: sha256:32964eaf382045310dab2281fddca478efe6f87c94ad5c6c7f055f5eb14c4ed2
Deleted: sha256:d4fc045c9e3a848011de66f34b81f052d4f2c15a17bb196d637e526349601820
PS C:\Users\bsulj>
```

Aufgabe 2:

1. Erzeuge einen neuen Node.js-Container namens test-container auf Grundlage des Standard-Node.js-Images und führe darin die bash aus

```
PS C:\Users\bsulj> docker run -it --name test-container node bash
Unable to find image 'node:latest' locally
latest: Pulling from library/node
7bb465c29149: Downloading [=====>                                             ]  5.585MB/49.55MB
2b9b41aaa3c5: Downloading [=======>                                           ]  3.697MB/24.05MB
49b40be4436e: Downloading [==>                                                ]  3.234MB/64.14MB
c558fac597f8: Waiting
449619e06fe3: Waiting
586631916d9c: Waiting
8020d38d3250: Waiting
38b1ded32c48: Waiting
```

2. Dieser Container soll nach Beenden NICHT automatisch gelöscht werden

   Damit das nicht passiert sollte der –rm Befehl nicht angeben werden

3. Steuere mit der bash im Container den Ordner etc an und finde die Version der zugrunde liegenden Debian-Distribution heraus

```
root@05231a1e0b4f:/# cd /etc
root@05231a1e0b4f:/etc# cat debian_version
12.5
root@05231a1e0b4f:/etc#
```

4. Gehe dann zurück in den root-Ordner und von dort zu /usr/share

```
root@05231a1e0b4f:/# cd /usr/share/
root@05231a1e0b4f:/usr/share# ls
```

5. Findest du in dem Ordner Hinweise zu einer anderen Programmiersprache, die wir schon verwendet haben und die in diesem Container installiert ist?

```
root@05231a1e0b4f:/usr/share# ls
ImageMagick-6  base-passwd      djvu        fonts     icons        locale        perl          tabset
X11            bash-completion  doc         gcc       icu          man           perl5         terminfo
aclocal        binfmts          doc-base    gdb       info         menu          pixmaps       thumbnailers
aclocal-1.16   bug              dpkg        gettext   keyrings     mercurial     pkgconfig     ucf
applications   ca-certificates  emacs       gir-1.0   libc-bin     mime          polkit-1      util-linux
apport         common-licenses  et          git-core  libgcrypt20  misc          python3       vala
autoconf       debconf          file        gitweb    libthai      mysql-common  readline      xml
automake-1.16  debianutils      fish        glib-2.0  libtool      pam           sensible-utils  zoneinfo
base-files     dict             fontconfig  gnupg     lintian      pam-configs   sgml          zsh
root@05231a1e0b4f:/usr/share#
```

Aufgabe 3:

1. Benenne dann den Container test-container um in my-node-app

```
exit
PS C:\Users\bsulj> docker rename test-container my-node-app
PS C:\Users\bsulj>
```

2. Erstelle dann dort dann den Ordner /app (Befehl: mkdir /app)

```
PS C:\Users\bsulj> docker exec -it my-node-app mkdir /app
PS C:\Users\bsulj>
```

3. Wechsel anschließend in diesen Ordner und erstelle eine Datei "main.js" mit folgendem Inhalt console.log("Hallo Welt")

```
PS C:\Users\bsulj> docker exec -it my-node-app bash
root@05231a1e0b4f:/# cd /app/
root@05231a1e0b4f:/app# echo "console.log('Hallo Welt')" > main.js
root@05231a1e0b4f:/app#
```

4. Führe anschließend dein Skript via node main.js aus

```
root@05231a1e0b4f:/app# node main.js
Hallo Welt
root@05231a1e0b4f:/app#
```