



Nama: **Bilhaq Avi Dewantara (120140141)**
Mata Kuliah: **Sistem Operasi (IF2223)**

Tugas Ke: **03**
Tanggal: **01/05/2022**

Daftar Anggota Kelompok	
NIM	Nama
120140141	Bilhaq Avi Dewantara
120140147	Gery Melia Suwanda
120140150	Fadhilah Fauza Hamda

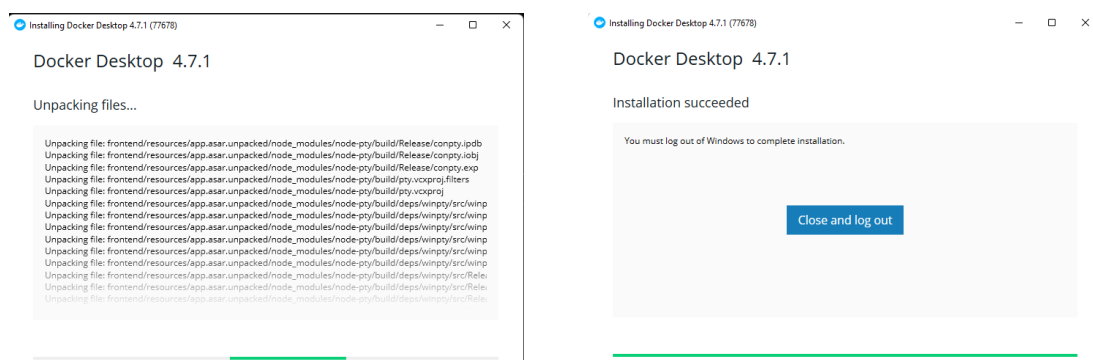
1 Tujuan Hands On 3

Tujuan adanya tugas Hands On 3 adalah memahami bagaimana cara kerja dari *Docker* ini dan mengimplementasikannya menggunakan *container* layaknya *cloud computing*. Dengan adanya *Docker*, kita dapat melakukan pembuatan aplikasi dengan mudah, karena *Docker* sudah terdapat *virtual machine* sebagai pembangun sebuah server. Oleh karena itu, kita tidak perlu membuat sebuah *virtual machine* sendiri untuk men-*deploy* aplikasi ke server.

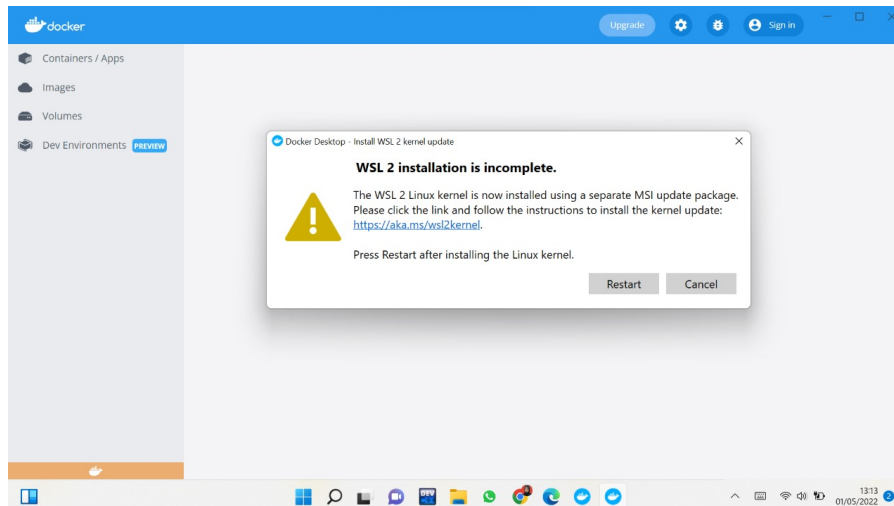
2 Setup Docker

Pertama yang harus dilakukan dalam Hands On 3 ini adalah mendownload *Docker*. Akan tetapi, hal tersebut tidak bisa dijalankan karena diperlukan *Requirements* yang akan dijelaskan sebagai berikut.

2.1 Requirements (WSL 2)



Gambar 1: Menginstal Docker

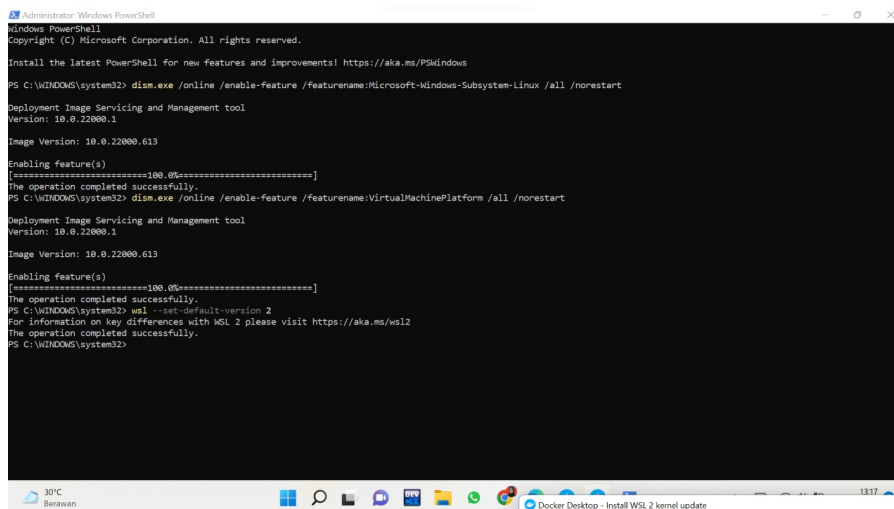


Gambar 2: Pop Up Membutuhkan WSL 2 Untuk Menjalankan Docker

WSL 2 atau *Windows Subsystem for Linux* adalah sistem operasi yang dikembangkan oleh Microsoft agar sistem operasi Windows dapat menjalankan aplikasi yang berbasis *Linux*. Dalam menginstall WSL 2 kita menggunakan Terminal Windows dengan *command* berikut.

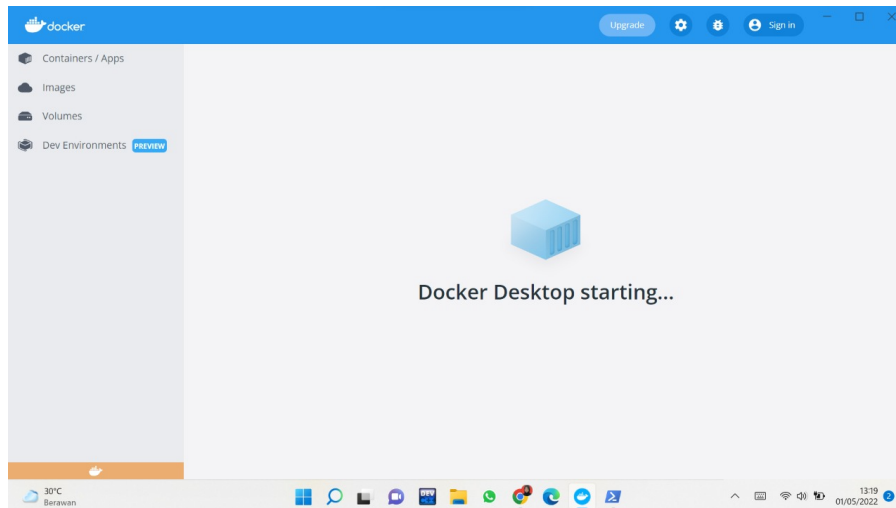
```
1 wsl --install Ubuntu # Menginstall WSL untuk distro Ubuntu
```

Atau terdapat alternatif lainnya dalam menginstall WSL 2 ini dengan menggunakan [Instalasi Manual](#) terlebih dahulu. Berikut merupakan gambar dari Instalasi Manual tersebut dari implementasi step 1 hingga step 5.



Gambar 3: Instalasi Manual WSL 2 Lewat Terminal Windows

Setelah di WSL 2 terinstal, maka kita harus me-*restart* PC kita terlebih dahulu. Setelah itu, barulah kita bisa membuka aplikasi *Docker* dan untuk mengetahui apakah *Docker* sudah berjalan atau belum ialah dengan melihat statusnya di sebelah bawah bagian kiri aplikasinya. Ketika sudah berwarna hijau maka *Docker* sudah bisa digunakan.



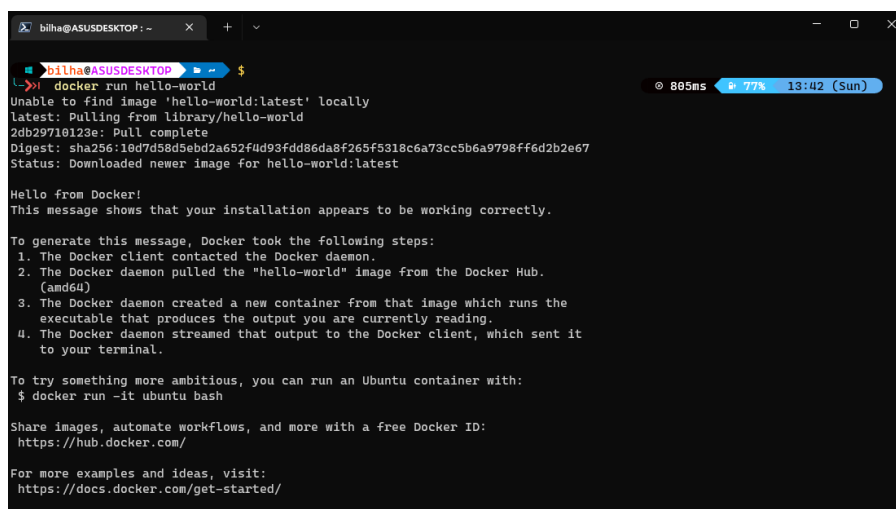
Gambar 4: Kondisi Docker Sedang Starting

3 Menjalankan Container Pertama

3.1 Hello-World

Pada kesempatan ini, kita sudah bisa memulai menjalankan perintah *Docker* dengan menggunakan PowerShell atau CommandPrompt sesuai dengan keinginan *User*. Awal mula yang harus kita coba untuk mengeceknya dengan mengetik perintah berikut pada PowerShell.

```
1 docker run hello-world
2
```



Gambar 5: Docker Run Hello-World

3.2 Alpine Linux

Pada langkah selanjutnya, kita akan mencoba menjalankan *container* dari *Alpine Linux* yang sudah terinstal di PC kita. Perintah yang harus digunakan pada langkah ini ialah melakukan *pull container* dengan perintah sebagai berikut.

```
1 docker pull alpine
2
```

```
bilha@ASUSDESKTOP ~$ docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
df9b9388f04a: Pull complete
Digest: sha256:4edbd2beb5f78b1014028f4fbb99f3237d9561100b6881aabbf5acce2c4f9454
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
```

Gambar 6: Docker Pull Alpine

3.2.1 Docker Images

Setelah langkah *pull container* berhasil dijalankan, yang mana akan memuat *library/alpine* ke dalam PC lalu disimpan ke dalam sistem yang akan digunakan. Langkah selanjutnya ialah mengecek image apa saja yang sudah pernah kita *pull* atau unduh dengan menggunakan perintah berikut.

```
1 docker images
```

```
bilha@ASUSDESKTOP ~$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
alpine latest 0ac33e5f5afa 3 weeks ago 5.57MB
hello-world latest feb5d9fea6a5 7 months ago 13.3kB
```

Gambar 7: Docker Images

Langkah berikutnya ialah kita akan memperlihatkan sistem memproses *container* yang telah terisolasi. Dalam hal ini, *Container* merupakan sebuah proses yang menjalankan *Host* yang mana *Host* berupa *local*. Saat operator *docker run* di eksekusi, *container* yang sudah dijalankan akan terisolasi akibat mempunyai file sistem dan jaringan sendiri dengan pohon proses yang terpisah dari *Host*. Untuk menjalankan *container docker alpine* dengan urutan *image* ini, kita dapat menggunakan perintah berikut.

```
1 docker run alpine ls -l
```

```
bilha@ASUSDESKTOP ~$ docker run alpine ls -l
total 56
drwxr-xr-x 2 root root 4096 Apr 4 16:06 bin
drwxr-xr-x 5 root root 340 May 1 06:57 dev
drwxr-xr-x 1 root root 4096 May 1 06:57 etc
drwxr-xr-x 2 root root 4096 Apr 4 16:06 home
drwxr-xr-x 7 root root 4096 Apr 4 16:06 lib
drwxr-xr-x 5 root root 4096 Apr 4 16:06 media
drwxr-xr-x 2 root root 4096 Apr 4 16:06 mnt
drwxr-xr-x 2 root root 4096 Apr 4 16:06 opt
dr-xr-xr-x 254 root root 0 May 1 06:57 proc
drwx----- 2 root root 4096 Apr 4 16:06 root
drwxr-xr-x 2 root root 4096 Apr 4 16:06 run
drwxr-xr-x 2 root root 4096 Apr 4 16:06 sbin
drwxr-xr-x 2 root root 4096 Apr 4 16:06 srv
dr-xr-xr-x 11 root root 0 May 1 06:57 sys
drwxrwxrwt 2 root root 4096 Apr 4 16:06 tmp
drwxr-xr-x 7 root root 4096 Apr 4 16:06 usr
drwxr-xr-x 12 root root 4096 Apr 4 16:06 var
```

Gambar 8: Docker Run Alpine ls -l

Perintah selanjutnya kita akan menampilkan "Hello World" pada Terminal dengan menggunakan perintah *container alpine docker* dan ditambah oleh perintah *echo* yang juga terdapat pada *container alpine*. Untuk perintahnya ialah sebagai berikut.

```
1 docker run alpine echo "Hello World"
```

```
bilha@ASUSDESKTOP ~$ docker run alpine echo "hello from alpine"
hello from alpine
```

Gambar 9: Hello World Pada Alpine

Berikutnya ialah kita mencoba masuk ke dalam *bash* dari *Container Alpine* tersebut dengan menambahkan opsi ketika menjalankan perintah *Cointainer* tersebut. Perintah yang digunakan ialah sebagai berikut.

```
1 docker run -it alpine /bin/sh
```

```
bilha@ASUSDESKTOP ~$ docker run -it alpine /bin/sh
/ #
```

Gambar 10: Bash dari Container Alpine

Dari Gambar 10 dapat terlihat bahwa tidak menampilkan *ouput* ke layar, karena hal tersebut hanya mengeluarkan *bash* atau *interactive shells* sesaat perintah di atas dijalankan. Oleh karena itu, diberikanlah *command* atau perintah baru di dalam */bin/sh* ini. Perintah yang akan kita berikan untuk menampilkan *list* dari seluruh file yang ada pada */bin/sh* ialah sebagai berikut.

```
1 / # ls
```

```
bilha@ASUSDESKTOP ~$ docker run -it alpine /bin/sh
/ # ls
bin  etc  lib  mnt  proc  run  srv  tmp  var
dev  home media opt  root  sbin sys  usr
```

Gambar 11: List dari /bin/sh

Selanjutnya masih pada *bash* *bin/sh*, kita akan memberikan perintah untuk menampilkan seluruh informasi dasar yang dimiliki oleh sistem. Perintah yang digunakan ialah sebagai berikut.

```
1 / # uname -a
```

```
bilha@ASUSDESKTOP ~ $
❯ docker run -it alpine /bin/sh
/ # uname -a
Linux 3f976ac3bb3c 5.10.16.3-microsoft-standard-WSL2 #1 SMP Fri Apr 2 22:23:49 UTC 2021 x86_64 Linux
```

Gambar 12: `uname -a` dari `/bin/sh`

Setelah kedua perintah pada `bin/sh` ini dijalankan maka kita bisa keluar dari *bash* tersebut dengan perintah.

```
1 / # exit
2
```

Pada langkah terakhir yang digunakan untuk menampilkan *container* yang sudah kita jalankan sebelumnya dengan menggunakan perintah.

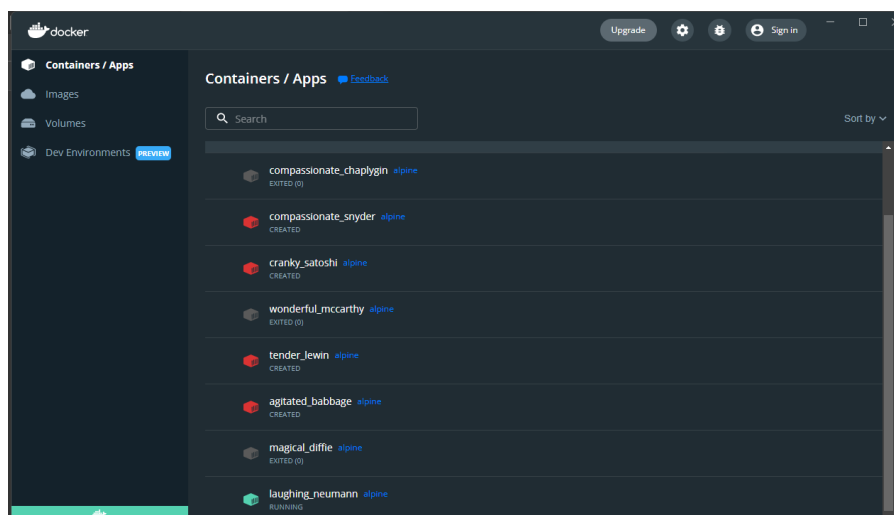
```
1 docker ps -a
2
```

```
bilha@ASUSDESKTOP ~ $
❯ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
30c9cab405e8	alpine	"/bin/sh"	4 minutes ago	Exited (0) 6 seconds ago		laughing_neumann
3f976ac3bb3c	alpine	"/bin/sh"	9 minutes ago	Exited (0) 7 minutes ago		magical_diffie
1fff1e	alpine	"-it /bin/sh"	10 minutes ago	Created		agitated_babbage
7db46f6d0e00	alpine	"it /bin/sh"	10 minutes ago	Created		tender_lewin
2d4b346ale61	alpine	"it /bin/sh"	10 minutes ago	Created		wonderful_mccarthy
win	alpine	"/bin/sh"	10 minutes ago	Exited (0) 10 minutes ago		cranky_satoshi
09f3be7adfef	alpine	"echo 'hello from al..."	14 minutes ago	Exited (0) 14 minutes ago		compassionate_snyder
_mccarthy	alpine	"it /bin/sh"	11 minutes ago	Created		intelligent_engelbart
e813bc42e4a9	alpine	"ls -l"	15 minutes ago	Exited (0) 15 minutes ago		interesting_grothendieck
toshi	alpine	"-it /bin/sh"	11 minutes ago	Created		
9583e92cfe98	alpine	"echo 'hello from al..."	14 minutes ago	Exited (0) 14 minutes ago		
nate_snyder	alpine	"echo 'hello from al..."	14 minutes ago	Exited (0) 14 minutes ago		
1f73b69095dd	alpine	"echo 'hello from al..."	14 minutes ago	Exited (0) 14 minutes ago		
nate_chaplygin	alpine	"echo 'hello from al..."	14 minutes ago	Exited (0) 14 minutes ago		
efef1bba459b	alpine	"ls -l"	15 minutes ago	Exited (0) 15 minutes ago		
nt_engelbart	alpine	"ls -l"	15 minutes ago	Exited (0) 15 minutes ago		
12d9a7937825	hello-world	"/hello"	28 minutes ago	Exited (0) 28 minutes ago		
ng_grothendieck	hello-world	"/hello"	28 minutes ago	Exited (0) 28 minutes ago		

Gambar 13: `Docker Ps -a`

Faktanya, kita dapat melihat *container* yang telah kita jalankan sebelumnya pada aplikasi *Docker* seperti pada gambar berikut.

Gambar 14: *Container* Pada Aplikasi Docker

4 Pertanyaan Hands On

4.1 Apa itu Docker?

Docker adalah sebuah aplikasi yang bersifat *opensource* yang memiliki fungsi sebagai wadah atau *container* untuk memasukkan sebuah perangkat lunak secara lengkap beserta semua hal lainnya yang dibutuhkan oleh perangkat lunak tersebut agar dapat berfungsi. Docker *container* ini hampir mirip seperti virtual machine, tetapi dengan sistem yang lebih ringan. Karena sistem *docker* tidak membawa keseluruhan sistem operasi [1].

4.2 Apa fungsi dari perintah "docker run"?

Perintah "*docker run*" berfungsi dalam menjalankan sebuah *container*. Dengan perintah ini, *docker* akan mencari *image* sesuai dengan letak lokal dari *container* tersebut dan menjalankannya. Apabila pada lokal tidak terdapat *image* yang diminta, maka akan memulai mencarinya di global dan akan melakukan *pull image* tersebut pada lokal.

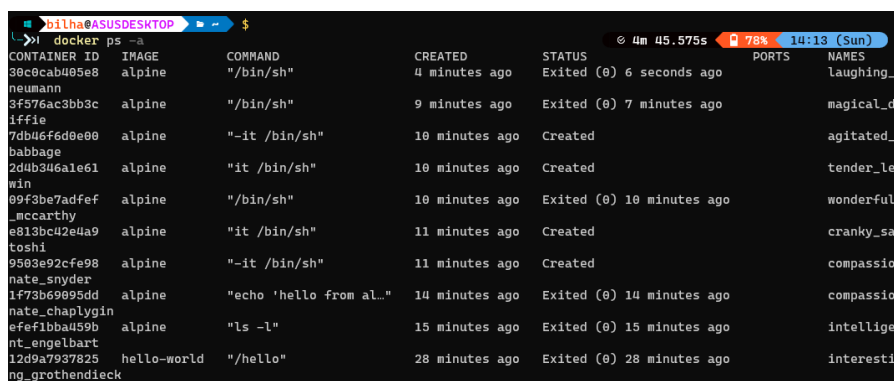
4.3 Apa yang dimaksud dengan *container*?

Container adalah wadah yang digunakan untuk mengemas dan menjalankan sebuah aplikasi. Wadah ini sendiri mencakup kode, *runtime*, *system tools*, dan pengaturan. Namun, container hanya dapat mengakses *resource* yang telah ditentukan dalam gambar *docker* saja.

4.4 Apa yang terjadi ketika perintah "docker ps -a" dilakukan?

Perintah *docker ps -a* adalah sebuah *command* untuk menampilkan data *container* apa saja yang pernah dibuat. Pada penerapannya, "ps -a" akan menampilkan dari *id* hingga ke nama *ports*-nya seperti gambar di bawah ini :

```
1 docker ps -a
2
```



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
38c0cab405e8	alpine	"/bin/sh"	4 minutes ago	Exited (0) 6 seconds ago		laughing_
neumann	alpine	"/bin/sh"	9 minutes ago	Exited (0) 7 minutes ago		magical_d
3f576ac3bb3c	alpine	"/bin/sh"	10 minutes ago	Created		agitated_
7db45f6d0e00	alpine	"-it /bin/sh"	10 minutes ago	Created		tender_le
babbage	alpine	"it /bin/sh"	10 minutes ago	Created		wonderful
2dub346ale61	alpine	"/bin/sh"	10 minutes ago	Created		cranky_sa
min	alpine	"it /bin/sh"	11 minutes ago	Created		compassio
09f3be7adfef	alpine	"echo 'hello from al..."	14 minutes ago	Exited (0) 14 minutes ago		compassio
mccarthy	alpine	"ls -l"	15 minutes ago	Exited (0) 15 minutes ago		intellige
e813bc42e4a9	alpine	"/hello"	28 minutes ago	Exited (0) 28 minutes ago		interesti
toshi	hello-world	"/bin/sh"				
9503e92cfe98						
nate_snyder						
1f73b69095dd						
nate_chaplygin						
efef1bbad59b						
nt_engelbart						
12d9a7937825						
ng_grothendieck						

Gambar 15: Docker Ps -a

4.5 Apa yang terjadi ketika perintah "docker run -it" dilakukan?

Docker run -it adalah sebuah *command* yang memberi instruksi kepada *docker* agar mengalokasi sebuah *pseudo-TTY* atau bentuk komunikasi dua arah antara komputer utama dengan ruang yang terisolasi dengan membuat sebuah *interactive bash shell* pada *container*.

```
1 docker run -it
2
```



Gambar 16: Docker Run -it

4.6 Apa yang dimaksud dengan Images?

Docker images adalah sebuah *blueprint* atau dasar dari aplikasi berbasis *docker* yang memiliki sifat *read-only*. Docker images sendiri memiliki fungsi untuk membuat *docker container*, yang mana dengan satu *docker images* dapat dibuat banyak *docker container* [2].

4.7 Apa yang dimaksud dengan *daemon*?

Daemon pada *docker* adalah sebuah program yang bertugas menyelesaikan tugas di belakang layar. Daemon ini berfungsi untuk men-*handling* dari *docker images*, *container*, volume, dan *network* di dalamnya pada *request services* secara berkala ketika sistem komputer menerima sebuah perintah. Selanjutnya *request* tersebut di *forward* menuju program lain yang umumnya berjalan di latar belakang [3].

5 Kesimpulan

Pada tugas Hands On 3 ini dapat kami simpulkan bahwa *Docker* merupakan aplikasi yang berfungsi dalam mempermudah seorang *developer* dalam melakukan *build*, *shipping*, dan *run* aplikasi pada dengan sebuah *container* melalui infrastruktur dari PC *developer* tersebut. Dengan demikian, *resource* yang diperlukan dalam pembuatan suatu aplikasi hanya sedikit saja dan itu mempermudah *developer*.

6 Link GitHub

Link GitHub dari Hands On 3 ini : [Klik disini](#)

References

- [1] S. Goasguen, “Docker cookbook,” 2015.
- [2] M. Hanif, “Sekilas tentang docker,” Jun 2017, accessed: 2022-05-01. [Online]. Available: <https://hanifmu.com/posts/sekilas-tentang-docker/>
- [3] T. Contributor, “What is daemon? - definition from whatis.com,” Sep 2005. [Online]. Available: <https://www.techtarget.com/whatis/definition/daemon>