

# **Buku Panduan Docker** <sup>1</sup>

## Docker <sup>2</sup>

CLASSROOM DEV TEAM<sup>3</sup>

August 29, 2022

<sup>1</sup>This is a footnote.

<sup>2</sup>This is yet another footnote.

<sup>3</sup>[www.example.com](http://www.example.com)



# Contents

<b>1 Tentang Docker</b>	<b>1</b>
1.1 Pengertian Docker . . . . .	1
1.2 Keuntungan Menggunakan Docker . . . . .	2
1.3 Arsitektur Docker . . . . .	2
1.3.1 Docker Daemon . . . . .	2
1.3.2 Docker Client . . . . .	2
1.3.3 Docker Desktop . . . . .	3
1.3.4 Docker Registries . . . . .	3
1.3.5 Docker CE . . . . .	3
1.3.6 Docker EE . . . . .	3
1.3.7 Docker Object . . . . .	4
<b>2 instalasi Docker di Server</b>	<b>5</b>
2.1 Pembuka . . . . .	5
2.2 Langkah Instalasi . . . . .	5
<b>3 Build Aplikasi dan Run Pada Docker</b>	<b>11</b>
3.1 Pembuka . . . . .	11
3.2 Langkah Build Aplikasi . . . . .	12
3.3 Run Pada Docker . . . . .	15
<b>4 Upload dan Pull Pada Image</b>	<b>17</b>
4.1 Pembuka . . . . .	18
4.2 Upload Image ke Docker Hub . . . . .	18
4.3 Pull Image dari Docker Registry . . . . .	22
<b>5 Docker Volume</b>	<b>23</b>
5.1 Pembuka . . . . .	23
5.2 Kapan menggunakan volume ? . . . . .	23
5.3 Membuat docker Volume . . . . .	24
<b>6 Memanipulasi File Pada Docker volume</b>	<b>27</b>
6.1 Pembuka . . . . .	27
6.2 Langkah memanipulasi file pada docker volume . . . . .	27

<b>7 Docker Network</b>	<b>35</b>
7.1 Pengertian . . . . .	35
7.2 Docker Network Drivers . . . . .	35
7.2.1 Bridge (default) . . . . .	35
7.2.2 Host . . . . .	35
7.2.3 Overlay . . . . .	36
7.2.4 IPvlan . . . . .	36
7.2.5 Macvlan . . . . .	36
7.2.6 None . . . . .	36
7.2.7 Network Plugins . . . . .	36
<b>8 Menghubungkan Lebih Dari 1 Aplikasi Dengan Docker Network</b>	<b>39</b>
8.1 Pembuka . . . . .	39
8.2 Langkah menghubungkan lebih dari 1 aplikasi dengan docker network . . . . .	39
<b>9 Perbedaan Docker Version</b>	<b>45</b>
9.1 Pembuka . . . . .	45
9.2 Perbedaan docker version . . . . .	46
<b>10 Tentang Docker Hub</b>	<b>49</b>
10.1 Pengertian . . . . .	49
10.2 Fitur Docker Hub . . . . .	49

# List of Figures

1.1	Docker . . . . .	1
1.2	Arsitektur Docker . . . . .	3
2.1	Apt update . . . . .	6
2.2	Apt menggunakan paket HTTPS . . . . .	6
2.3	Tambahkan kunci GPG . . . . .	7
2.4	Tambahkan repositori docker . . . . .	7
2.5	Update paket . . . . .	8
2.6	Repositori instalasi docker dari repositori docker . . . . .	8
2.7	Instalasi docker . . . . .	9
2.8	Memastikan docker berjalan . . . . .	9
3.1	Buat folder dan buat dockerfile didalamnya . . . . .	12
3.2	Isi file dockerfile . . . . .	13
3.3	Build Image . . . . .	13
3.4	Cek daftar docker images . . . . .	14
3.5	Tag docker images . . . . .	14
3.6	Buat dan run images . . . . .	15
3.7	Cek kontainer . . . . .	16
3.8	Tampilan akses ke kontainer . . . . .	16
4.1	Buat repositori docker hub . . . . .	18
4.2	Pengaturan repositori . . . . .	19
4.3	Berhasil membuat repositori . . . . .	19
4.4	Login docker di command line . . . . .	20
4.5	Upload docker images . . . . .	20
4.6	Tampilan berhasil upload . . . . .	21
4.7	Pull docker images . . . . .	22
5.1	Docker Volume . . . . .	24
5.2	Buat docker volume saat container dibuat . . . . .	24
5.3	Tampilan mengakses kontainer dalam volume . . . . .	25
5.4	Tampilan inspect docker volume . . . . .	25

6.1	Buat Dockerfile . . . . .	28
6.2	Buat docker images . . . . .	28
6.3	Beri nama dan tag images . . . . .	29
6.4	Buat volume, masuk direktori container . . . . .	29
6.5	Cek file mount . . . . .	30
6.6	Inspect volume . . . . .	31
6.7	Edit file . . . . .	32
6.8	Edit file dan simpan . . . . .	32
6.9	Cek isi file . . . . .	33
7.1	Lihat port jaringan . . . . .	37
8.1	Cek docker network . . . . .	40
8.2	Buat docker network . . . . .	40
8.3	Masukan container ke docker network . . . . .	41
8.4	Inspect docker network . . . . .	41
8.5	Jalankan kontainer . . . . .	42
8.6	Tambah container ke docker network saat berjalan . . . . .	42
8.7	Inspect docker network . . . . .	43
8.8	Memutuskan docker network . . . . .	43
8.9	Inspect docker network . . . . .	44
8.10	Menghapus docker network . . . . .	44
9.1	Cek versi docker . . . . .	46

# List of Tables



# 1

## Tentang Docker

*“Docker makes development efficient and predictable”*

– Docker, [docker.com](https://www.docker.com)

### 1.1 Pengertian Docker

Dikutip dari situs Docker Docs, Docker merupakan aplikasi open source untuk developing, shipping dan running aplikasi. Docker memungkinkan aplikasi untuk terpisah dari infrastrukturnya, sehingga proses delivery aplikasi dapat dilakukan dengan cepat.

Docker menyediakan kemampuan untuk mengemas dan menjalankan aplikasi di lingkungan yang terisolasi yang disebut container. Dengan konsep isolasi dan security docker pengguna dapat menjalankan banyak container secara bersamaan pada host tertentu. Container berisi semua yang diperlukan untuk menjalankan aplikasi, sehingga tidak perlu bergantung pada apa yang saat ini terinstall pada host. Selain itu pengguna dapat berbagi container dengan mudah.



Figure 1.1: Docker

## 1.2 Keuntungan Menggunakan Docker

Penggunaan docker mempunyai beberapa keuntungan bagi penggunanya baik developer ataupun perusahaan. Berikut merupakan keuntungan menggunakan docker :

1. Cepat dan konsisten dalam deliveri aplikasi
2. Kontainer docker dapat berjalan dimana saja, baik lokal, mesin fisik/virtual, cloud provider, dan lingkungan campuran lainnya
3. Menjalankan lebih dari 1 kontainer dalam satu perangkat yang sama
4. Ukuran kontainer yang relatif kecil
5. Hemat biaya dalam perangkat sehingga dengan spesifikasi hardware yang kurang namun banyak komputasi yang dijalankan

## 1.3 Arsitektur Docker

Dikutip dari situs Docker Docs, Docker menggunakan struktur client-server. Docker client dapat berkomunikasi dengan Docker daemon yang menjalankan tugas untuk membangun, menjalankan, dan mendistribusikan container docker. Docker client dan docker daemon dapat berjalan dalam sistem yang sama, ataupun secara remote/jarak jauh. docker client dan docker daemon berkomunikasi menggunakan REST API melalui socket UNIX atau interface jaringan. Docker client lainnya yaitu Docker Compose yang memungkinkan client untuk berkomunikasi dengan aplikasi yang didalamnya terdapat banyak container

### 1.3.1 Docker Daemon

Docker daemon/dockerd mendengarkan permintaan Docker API dan mengelola objek docker seperti images, container, networks dan volume. Docker daemon juga dapat berkomunikasi dengan daemon lain untuk mengelola layanan docker.

### 1.3.2 Docker Client

Docker client merupakan pengguna docker. Saat pengguna menjalankan perintah seperti run maka docker client akan mengirimkan perintah ini ke dockerd dan menjalankannya. Menggunakan docker API dan dapat berkomunikasi lebih dari satu daemon.

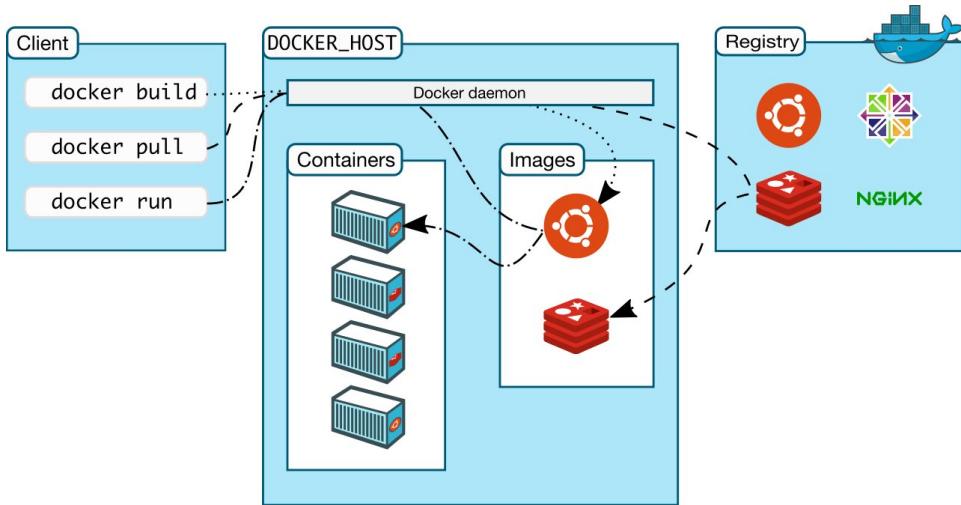


Figure 1.2: Arsitektur Docker

### 1.3.3 Docker Desktop

Docker desktop merupakan aplikasi versi GUI dari docker untuk pengguna yang ingin berinteraksi dengan docker lewat tampilan.

### 1.3.4 Docker Registries

Docker registries merupakan registri publik untuk menyimpan docker images yang dapat diakses oleh publik. Pengguna juga dapat membuat registri/repositori sendiri dan mengupload docker images pengguna ke registri docker. Saat menggunakan perintah docker pull atau docker run, docker image akan diambil dari registri yang dikonfigurasi, saat menggunakan perintah push docker images akan didorong/diupload ke registri yang dikonfigurasi. Docker registri yang dapat diakses secara publik yaitu docker Hub.

### 1.3.5 Docker CE

Docker CE atau Docker Community Edition adalah varian docker untuk developer dan small teams yang baru menggunakan docker dan melakukan eksperimen aplikasi berbasis container. Docker CE mempunyai 2 sumber update yaitu Stable dan Edge. Update stable yaitu update yang tersedia setiap tiga bulan sekali, dan Edge yaitu update setiap bulan.

### 1.3.6 Docker EE

Docker EE atau Docker Enterprise Edition adalah varian docker untuk enterprise development dan IT teams. Docker EE terintegrasi, bersertifikat,

dan didukung untuk menyediakan perusahaan dengan platform container paling aman di industri.

### 1.3.7 Docker Object

#### Images

Docker images merupakan template read-only yang digunakan untuk membangun container docker. Docker images dapat berisi images yang lain dan memuat detail konfigurasi yang diperlukan untuk membuat aplikasi berjalan dengan baik. Images dapat dibuat oleh pengguna ataupun menggunakan images yang diambil dari docker registries. Jika images dibuat oleh pengguna sendiri, maka pengguna membuat Dockerfile yang berisi sintaks dan langkah yang diperlukan untuk menjalankan aplikasi dalam images. Docker images lebih ringan, kecil dan cepat jika dibandingkan dengan teknologi virtualisasi lainnya.

#### Container

Container merupakan wadah/tempat untuk menjalankan aplikasi secara isolasi/terpisah dari lingkungan infrastruktur. Container berisikan docker images yang dapat dijalankan. Pengguna dapat membuat, memulai, menghentikan, memindahkan atau menghapus container menggunakan Docker API atau CLI. Dengan container pengguna dapat menjalankan lebih dari satu container dalam satu host, kemudahan dalam distribusi dan pengujian aplikasi, dan terintegrasi baik secara lokal, cloud, dan hybrid sekalipun.

# **2**

## **instalasi Docker di Server**

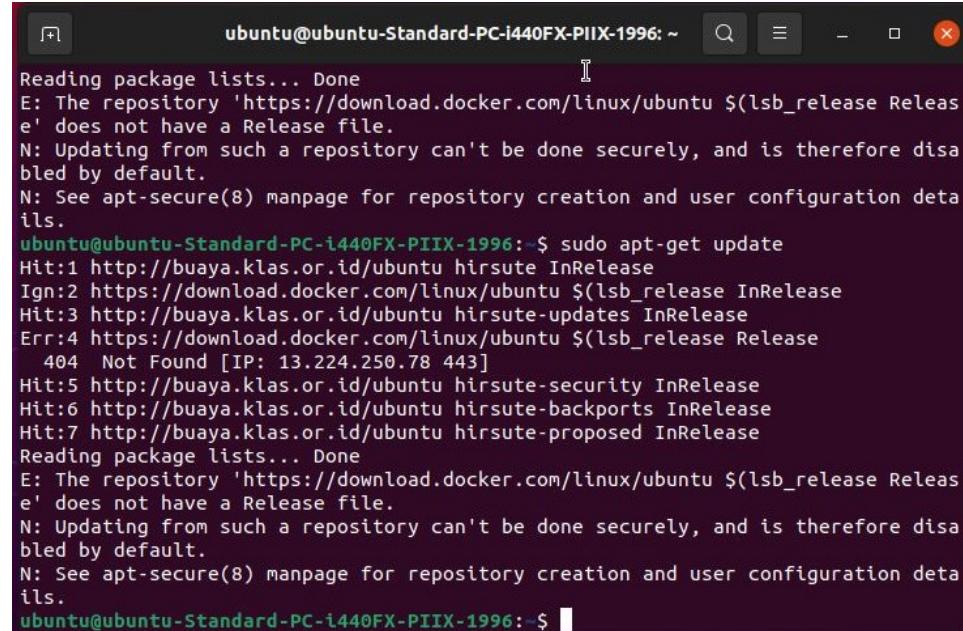
### **2.1 Pembuka**

Panduan kali ini membahas cara instalasi docker di server. Server yang digunakan kali ini menggunakan sistem operasi Linux Ubuntu 21.04 dengan versi docker terbaru saat didokumentasikan yaitu 20.10.17.

### **2.2 Langkah Instalasi**

### 1. Lakukan update paket pada linux

COMMAND : sudo apt-get update

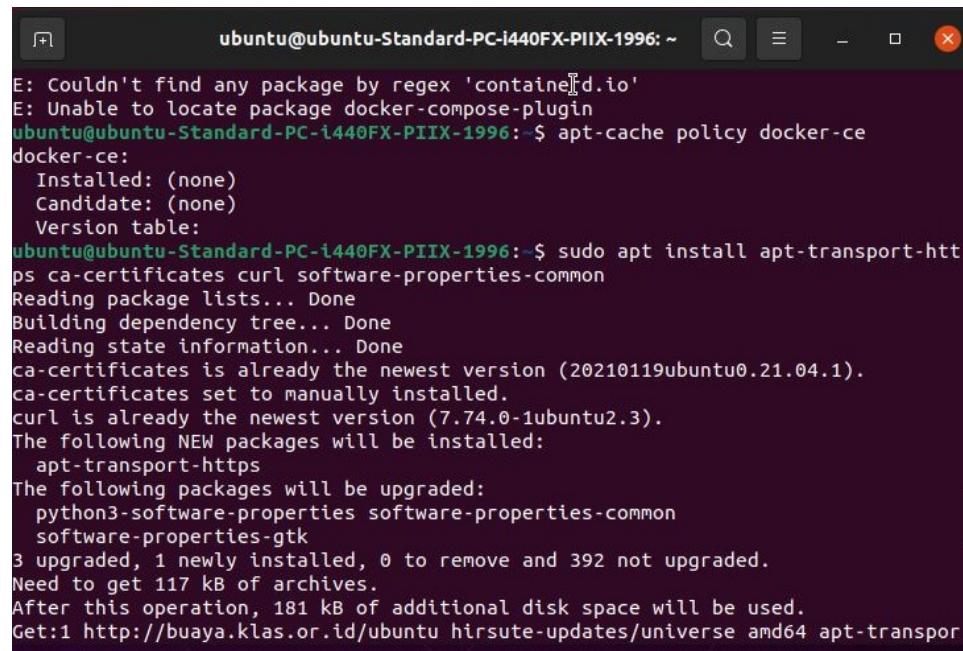


```
ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996: ~
Reading package lists... Done
E: The repository 'https://download.docker.com/linux/ubuntu $(lsb_release Releas
e' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disa
bled by default.
N: See apt-secure(8) manpage for repository creation and user configuration deta
ils.
ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996: ~$ sudo apt-get update
Hit:1 http://buaya.klas.or.id/ubuntu hirsute InRelease
Ign:2 https://download.docker.com/linux/ubuntu $(lsb_release InRelease
Hit:3 http://buaya.klas.or.id/ubuntu hirsute-updates InRelease
Err:4 https://download.docker.com/linux/ubuntu $(lsb_release Release
  404  Not Found [IP: 13.224.250.78 443]
Hit:5 http://buaya.klas.or.id/ubuntu hirsute-security InRelease
Hit:6 http://buaya.klas.or.id/ubuntu hirsute-backports InRelease
Hit:7 http://buaya.klas.or.id/ubuntu hirsute-proposed InRelease
Reading package lists... Done
E: The repository 'https://download.docker.com/linux/ubuntu $(lsb_release Releas
e' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disa
bled by default.
N: See apt-secure(8) manpage for repository creation and user configuration deta
ils.
ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996: ~$
```

Figure 2.1: Apt update

### 2. Install paket yang memungkinkan apt menggunakan paket dari HTTPS

COMMAND: sudo apt install apt-transport-https ca-certificates curl software-properties-common

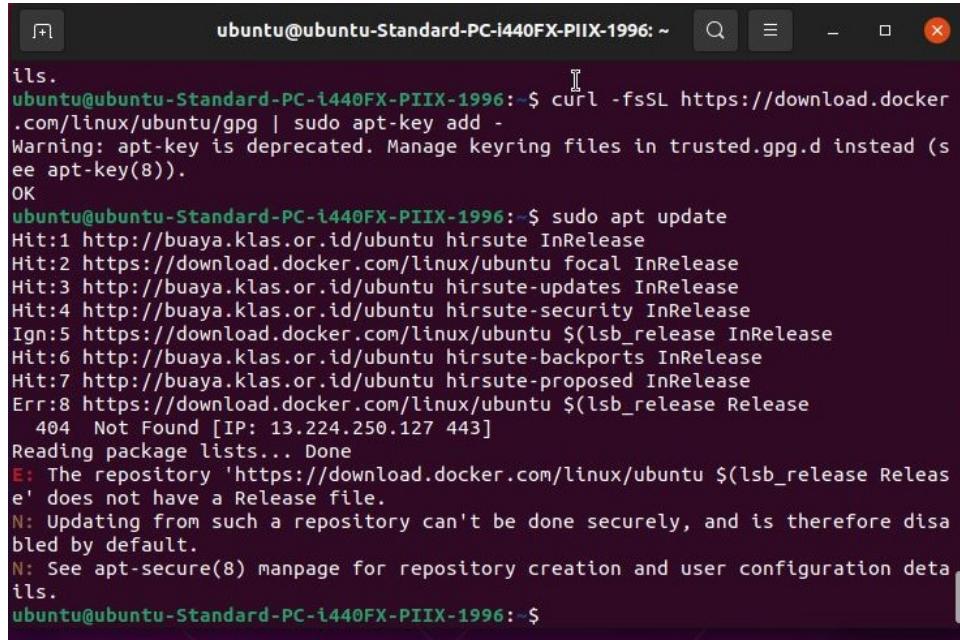


```
ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996: ~
E: Couldn't find any package by regex 'containerd.io'
E: Unable to locate package docker-compose-plugin
ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996: ~$ apt-cache policy docker-ce
docker-ce:
  Installed: (none)
  Candidate: (none)
  Version table:
ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996: ~$ sudo apt install apt-transport-htt
ps ca-certificates curl software-properties-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20210119ubuntu0.21.04.1).
ca-certificates set to manually installed.
curl is already the newest version (7.74.0-1ubuntu2.3).
The following NEW packages will be installed:
  apt-transport-https
The following packages will be upgraded:
  python3-software-properties software-properties-common
    software-properties-gtk
3 upgraded, 1 newly installed, 0 to remove and 392 not upgraded.
Need to get 117 kB of archives.
After this operation, 181 kB of additional disk space will be used.
Get:1 http://buaya.klas.or.id/ubuntu hirsute-updates/universe amd64 apt-transport-
```

Figure 2.2: Apt menggunakan paket HTTPS

### 3. Tambahkan kunci GPG untuk repositori resmi docker ke sistem dan lakukan update paket lagi

COMMAND: curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -  
sudo apt update

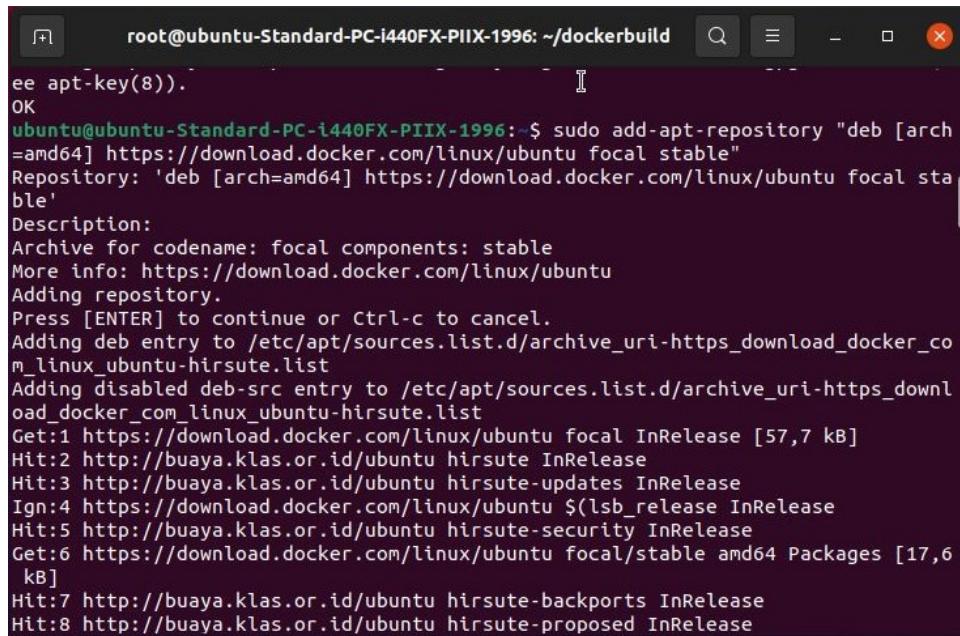


```
ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996: ~
ils.
ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996: ~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996: ~$ sudo apt update
Hit:1 http://buaya.klas.or.id/ubuntu hirsute InRelease
Hit:2 https://download.docker.com/linux/ubuntu focal InRelease
Hit:3 http://buaya.klas.or.id/ubuntu hirsute-updates InRelease
Hit:4 http://buaya.klas.or.id/ubuntu hirsute-security InRelease
Ign:5 https://download.docker.com/linux/ubuntu $(lsb_release InRelease
Hit:6 http://buaya.klas.or.id/ubuntu hirsute-backports InRelease
Hit:7 http://buaya.klas.or.id/ubuntu hirsute-proposed InRelease
Err:8 https://download.docker.com/linux/ubuntu $(lsb_release Release
  404 Not Found [IP: 13.224.250.127 443]
Reading package lists... Done
E: The repository 'https://download.docker.com/linux/ubuntu $(lsb_release Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996: ~$
```

Figure 2.3: Tambahkan kunci GPG

### 4. Tambahkan repositori docker

COMMAND: sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"

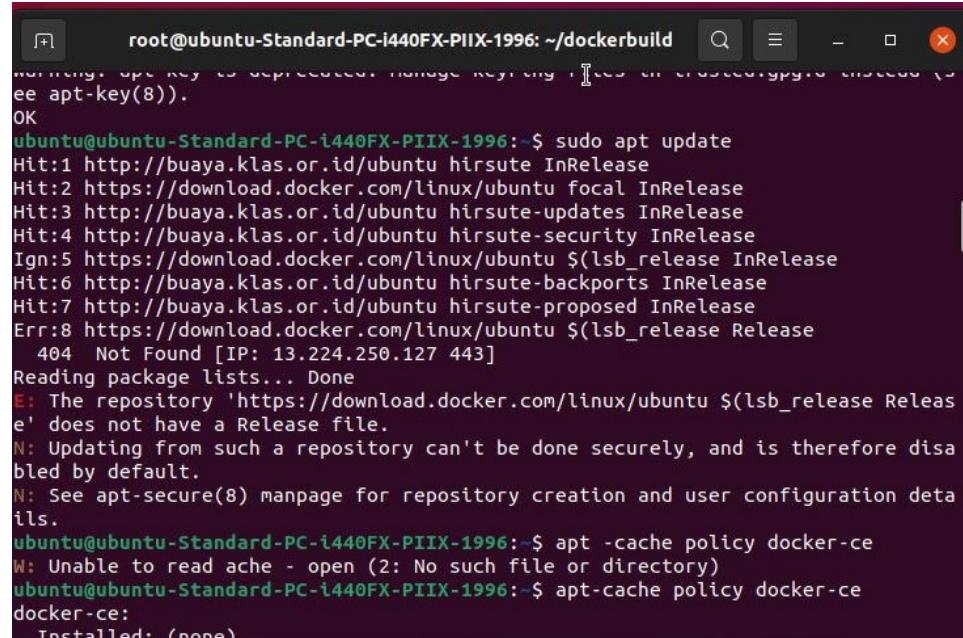


```
root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~/dockerbuild
ee apt-key(8)).
OK
ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996: ~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable'
Description:
Archive for codename: focal components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-hirsute.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-hirsute.list
Get:1 https://download.docker.com/linux/ubuntu focal InRelease [57,7 kB]
Hit:2 http://buaya.klas.or.id/ubuntu hirsute InRelease
Hit:3 http://buaya.klas.or.id/ubuntu hirsute-updates InRelease
Ign:4 https://download.docker.com/linux/ubuntu $(lsb_release InRelease
Hit:5 http://buaya.klas.or.id/ubuntu hirsute-security InRelease
Get:6 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages [17,6 kB]
Hit:7 http://buaya.klas.or.id/ubuntu hirsute-backports InRelease
Hit:8 http://buaya.klas.or.id/ubuntu hirsute-proposed InRelease
```

Figure 2.4: Tambahkan repositori docker

### 5. Lakukan update paket lagi

COMMAND: sudo apt update



```

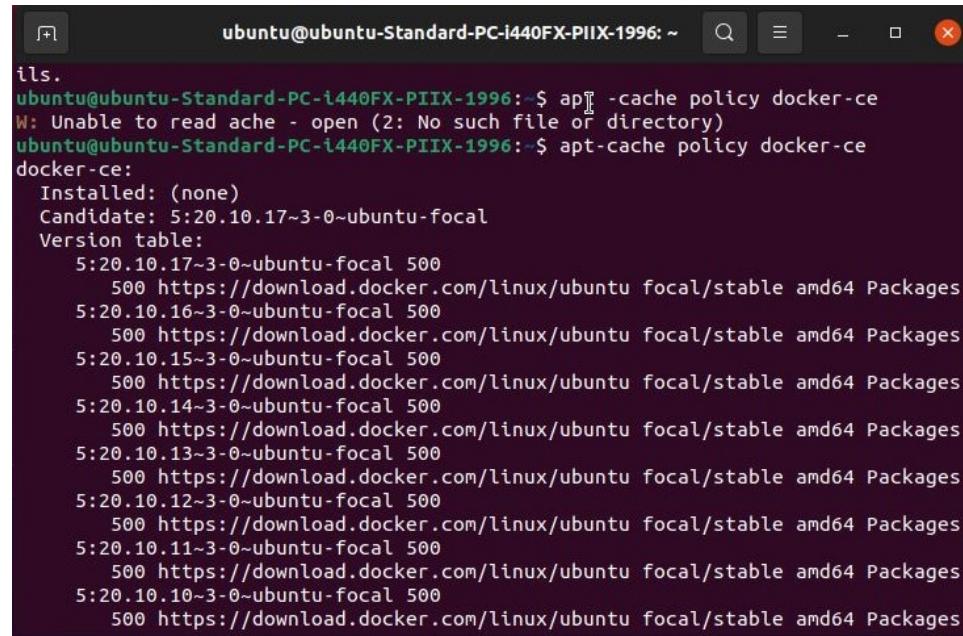
root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~/dockerbuild
Warning: apt key is deprecated. Manage keyring via /etc/apt/trusted.gpg.d instead of /etc/apt-key(8).
OK
ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996: $ sudo apt update
Hit:1 http://buaya.klas.or.id/ubuntu hirsute InRelease
Hit:2 https://download.docker.com/linux/ubuntu focal InRelease
Hit:3 http://buaya.klas.or.id/ubuntu hirsute-updates InRelease
Hit:4 http://buaya.klas.or.id/ubuntu hirsute-security InRelease
Ign:5 https://download.docker.com/linux/ubuntu $(lsb_release InRelease)
Hit:6 http://buaya.klas.or.id/ubuntu hirsute-backports InRelease
Hit:7 http://buaya.klas.or.id/ubuntu hirsute-proposed InRelease
Err:8 https://download.docker.com/linux/ubuntu $(lsb_release Release
      404  Not Found [IP: 13.224.250.127 443]
Reading package lists... Done
E: The repository 'https://download.docker.com/linux/ubuntu $(lsb_release Release
e' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996: $ apt -cache policy docker-ce
W: Unable to read cache - open (2: No such file or directory)
ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996: $ apt-cache policy docker-ce
docker-ce:
  Installed: (none)

```

Figure 2.5: Update paket

### 6. Pastikan repositori instalasi docker dari repositori docker bukan ubuntu

COMMAND: apt -cache policy docker-ce



```

ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996: ~
ils.
ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996: $ apt -cache policy docker-ce
W: Unable to read cache - open (2: No such file or directory)
ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996: $ apt-cache policy docker-ce
docker-ce:
  Installed: (none)
  Candidate: 5:20.10.17~3~0~ubuntu-focal
  Version table:
    5:20.10.17~3~0~ubuntu-focal 500
      500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
    5:20.10.16~3~0~ubuntu-focal 500
      500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
    5:20.10.15~3~0~ubuntu-focal 500
      500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
    5:20.10.14~3~0~ubuntu-focal 500
      500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
    5:20.10.13~3~0~ubuntu-focal 500
      500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
    5:20.10.12~3~0~ubuntu-focal 500
      500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
    5:20.10.11~3~0~ubuntu-focal 500
      500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
    5:20.10.10~3~0~ubuntu-focal 500
      500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages

```

Figure 2.6: Repositori instalasi docker dari repositori docker

## 7. Lakukan instalasi docker

COMMAND: `sudo apt install docker-ce`

```

root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~/dockerbuild
500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
5:19.03.9~3-0~ubuntu-focal 500
500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
ubuntu@ubuntu-Standard-PC-i440FX-PIIX-1996:~$ sudo apt install docker-ce
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-ce-cli docker-ce-rootless-extras docker-scan-plugin git
  git-man liberror-perl libslirp0 pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite git-daemon-run | git-daemon-sysvinit
  git-doc git-el git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  containerd.io docker-ce docker-ce-cli docker-ce-rootless-extras
  docker-scan-plugin git git-man liberror-perl libslirp0 pigz slirp4netns
0 upgraded, 11 newly installed, 0 to remove and 392 not upgraded.
Need to get 106 MB of archives.
After this operation, 453 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 https://download.docker.com/linux/ubuntu focal/stable amd64 containerd.io
amd64 1.6.6-1 [28,1 MB]
Get:2 http://buaya.klas.or.id/ubuntu hirsute/universe amd64 pigz amd64 2.6-1 [63
,6 kB]
```

Figure 2.7: Instalasi docker

## 8. Pastikan docker berjalan

COMMAND: `systemctl status docker`

```

root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockerbuild
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockerbuild# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2022-08-08 09:49:53 WIB; 1h 31min ago
     Docs: https://docs.docker.com
   Main PID: 8159 (dockerd)
      Tasks: 9
     Memory: 32.7M
        CPU: 0.000 CPU(s) since start
       CGroup: /system.slice/docker.service
               └─8159 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/cont>

Agu 08 09:49:51 ubuntu-Standard-PC-i440FX-PIIX-1996 dockerd[8159]: time="2022-08-08T09:49:51+00:00" level=info msg="Docker daemon <version> (channel 1.23.0) is starting as containerd interface"
Agu 08 09:49:52 ubuntu-Standard-PC-i440FX-PIIX-1996 dockerd[8159]: time="2022-08-08T09:49:52+00:00" level=info msg="Docker daemon <version> (channel 1.23.0) is running as containerd interface"
Agu 08 09:49:52 ubuntu-Standard-PC-i440FX-PIIX-1996 dockerd[8159]: time="2022-08-08T09:49:52+00:00" level=info msg="Docker daemon <version> (channel 1.23.0) is running as containerd interface"
Agu 08 09:49:53 ubuntu-Standard-PC-i440FX-PIIX-1996 dockerd[8159]: time="2022-08-08T09:49:53+00:00" level=info msg="Docker daemon <version> (channel 1.23.0) is running as containerd interface"
Agu 08 09:49:53 ubuntu-Standard-PC-i440FX-PIIX-1996 dockerd[8159]: time="2022-08-08T09:49:53+00:00" level=info msg="Docker daemon <version> (channel 1.23.0) is running as containerd interface"
Agu 08 09:49:53 ubuntu-Standard-PC-i440FX-PIIX-1996 systemd[1]: Started Docker daemon.
Agu 08 09:49:53 ubuntu-Standard-PC-i440FX-PIIX-1996 dockerd[8159]: time="2022-08-08T09:49:53+00:00" level=info msg="Docker daemon <version> (channel 1.23.0) is running as containerd interface"
Agu 08 09:57:45 ubuntu-Standard-PC-i440FX-PIIX-1996 dockerd[8159]: time="2022-08-08T09:57:45+00:00" level=info msg="Docker daemon <version> (channel 1.23.0) is running as containerd interface"
Agu 08 10:13:00 ubuntu-Standard-PC-i440FX-PIIX-1996 dockerd[8159]: time="2022-08-08T10:13:00+00:00" level=info msg="Docker daemon <version> (channel 1.23.0) is running as containerd interface"
Agu 08 10:13:00 ubuntu-Standard-PC-i440FX-PIIX-1996 dockerd[8159]: time="2022-08-08T10:13:00+00:00" level=info msg="Docker daemon <version> (channel 1.23.0) is running as containerd interface"
lines 1-21/21 (END)
```

Figure 2.8: Memastikan docker berjalan



# 3

## Build Aplikasi dan Run Pada Docker

### 3.1 Pembuka

Panduan kali ini membahas cara build aplikasi dan menjalankannya pada docker. Build yang dilakukan merupakan pembuatan docker images dari Dockerfile. Dockerfile merupakan file docker yang isinya konfigurasi yang diperlukan untuk menjalankan aplikasi. Setelah images dibuat dilakukan running aplikasi dengan menggunakan container.

## 3.2 Langkah Build Aplikasi

1. Buat folder baru, masuk direktori folder tersebut, dan buat file bernama Dockerfile

Buat folder baru bernama dockertest: `mkdir <NAMA FOLDER>`

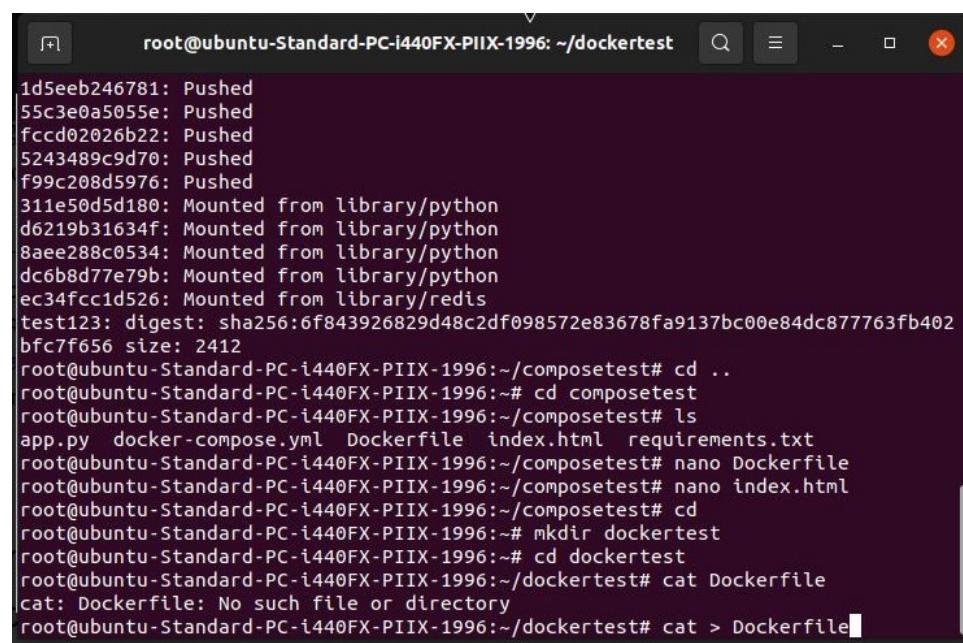
COMMAND: `mkdir dockertest`

Masuk direktori folder dockertest : `cd <NAMA FOLDER>`

COMMAND: `cd dockertest`

Buat file bernama Dockerfile:

COMMAND: `cat > dockertest`, Klik ENTER, klik CTRL+D



```

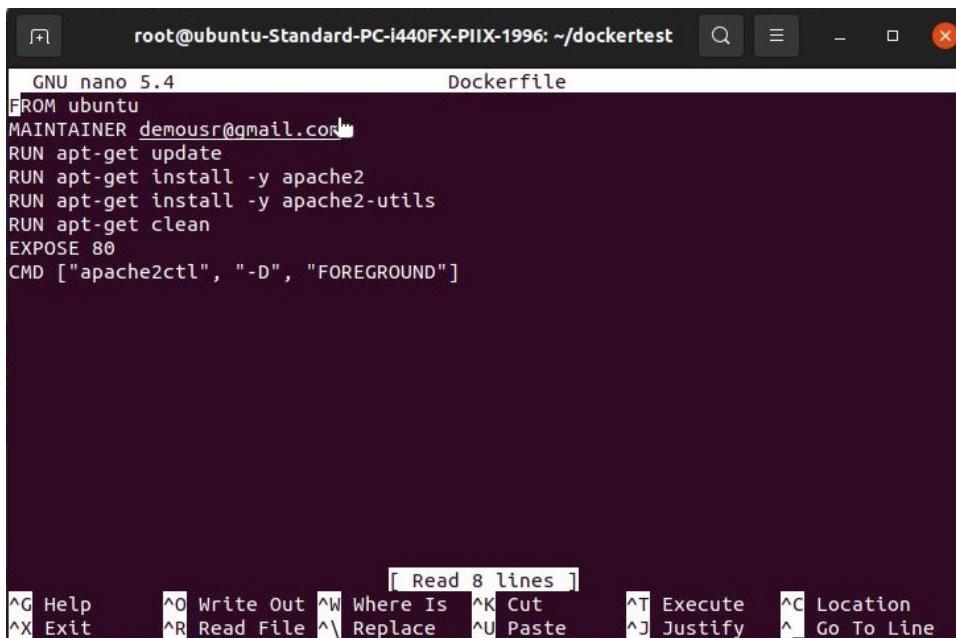
root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~/dockertest
1d5eeb246781: Pushed
55c3e0a5055e: Pushed
fccd02026b22: Pushed
5243489c9d70: Pushed
f99c208d5976: Pushed
311e50d5d180: Mounted from library/python
d6219b31634f: Mounted from library/python
8aee288c0534: Mounted from library/python
dc6b8d77e79b: Mounted from library/python
ec34fcc1d526: Mounted from library/redis
test123: digest: sha256:6f843926829d48c2df098572e83678fa9137bc00e84dc877763fb402
bfc7f656 size: 2412
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/composetest# cd ..
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# cd composetest
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/composetest# ls
app.py docker-compose.yml Dockerfile index.html requirements.txt
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/composetest# nano Dockerfile
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/composetest# nano index.html
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/composetest# cd ..
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# mkdir dockertest
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# cd dockertest
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# cat Dockerfile
cat: Dockerfile: No such file or directory
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# cat > Dockerfile

```

Figure 3.1: Buat folder dan buat dockerfile didalamnya

2. Edit file Dockerfile menggunakan editor nano dan isikan data seperti berikut

COMMAND: `nano Dockerfile`



```
GNU nano 5.4                               Dockerfile
FROM ubuntu
MAINTAINER demousr@gmail.com
RUN apt-get update
RUN apt-get install -y apache2
RUN apt-get install -y apache2-utils
RUN apt-get clean
EXPOSE 80
CMD ["apache2ctl", "-D", "foreground"]
```

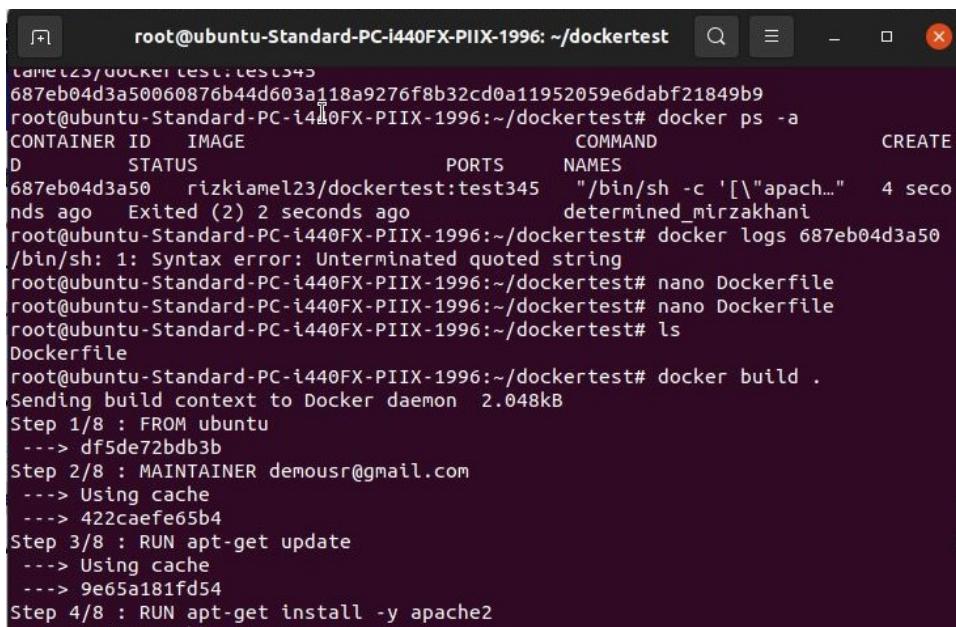
The screenshot shows a terminal window titled "root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~/dockertest". It displays a nano text editor with the Dockerfile content. The bottom of the screen shows nano key bindings: `[ Read 8 lines ]`, `^G Help`, `^O Write Out`, `^W Where Is`, `^K Cut`, `^T Execute`, `^C Location`, `^X Exit`, `^R Read File`, `^\\ Replace`, `^U Paste`, `^J Justify`, and `^A Go To Line`.

Figure 3.2: Isi file dockerfile

klik CTRL+X, klik Y, klik ENTER

3. Build image dari Dockerfile yang sudah dibuat

COMMAND: `docker build .`



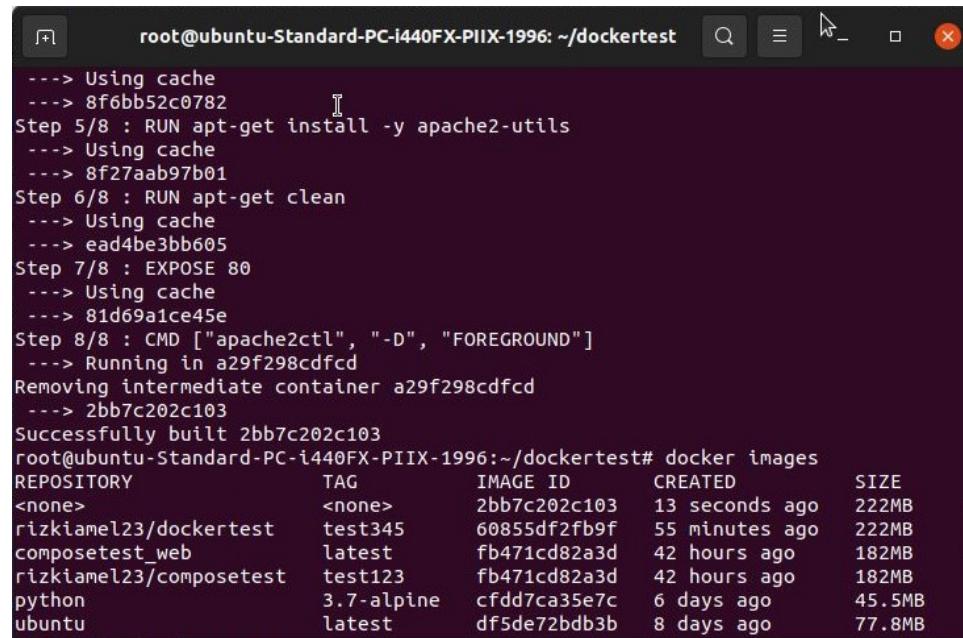
```
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
687eb04d3a50        rizkiamel23/dockertest:latest        "/bin/sh -c '[\"apach...'   4 seconds ago       Exited (2) 2 seconds ago          determined_mirzakhani
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker logs 687eb04d3a50
/bin/sh: 1: Syntax error: Unterminated quoted string
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# nano Dockerfile
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# nano Dockerfile
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# ls
Dockerfile
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker build .
Sending build context to Docker daemon 2.048kB
Step 1/8 : FROM ubuntu
--> df5de72bdb3b
Step 2/8 : MAINTAINER demousr@gmail.com
--> Using cache
--> 422caefef5b4
Step 3/8 : RUN apt-get update
--> Using cache
--> 9e65a181fd54
Step 4/8 : RUN apt-get install -y apache2
--> Using cache
```

The screenshot shows a terminal window titled "root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~/dockertest". It displays the output of the `docker build .` command. The terminal shows the creation of a container, its exit status, and the resulting image name. It also shows the log output of the container, the creation of a Dockerfile, and the build process itself, including cache hits for steps 2, 3, and 4.

Figure 3.3: Build Image

4. Apabila berhasil maka akan tampil pesan Successfull built <IMAGE ID>, lalu cek daftar docker images

COMMAND: `docker images`



```

root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest#
--> Using cache
--> 8f6bb52c0782
Step 5/8 : RUN apt-get install -y apache2-utils
--> Using cache
--> 8f27aab97b01
Step 6/8 : RUN apt-get clean
--> Using cache
--> ead4be3bb605
Step 7/8 : EXPOSE 80
--> Using cache
--> 81d69a1ce45e
Step 8/8 : CMD ["apache2ctl", "-D", "FOREGROUND"]
--> Running in a29f298cdfcd
Removing intermediate container a29f298cdfcd
--> 2bb7c202c103
Successfully built 2bb7c202c103
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker images
REPOSITORY          TAG      IMAGE ID      CREATED     SIZE
<none>            <none>   2bb7c202c103  13 seconds ago  222MB
rizkiamel23/dockertest  test345  60855df2fb9f  55 minutes ago  222MB
composetest_web    latest   fb471cd82a3d  42 hours ago   182MB
rizkiamel23/composetest  test123  fb471cd82a3d  42 hours ago   182MB
python              3.7-alpine  cfdd7ca35e7c  6 days ago    45.5MB
ubuntu              latest   df5de72bdb3b  8 days ago    77.8MB

```

Figure 3.4: Cek daftar docker images

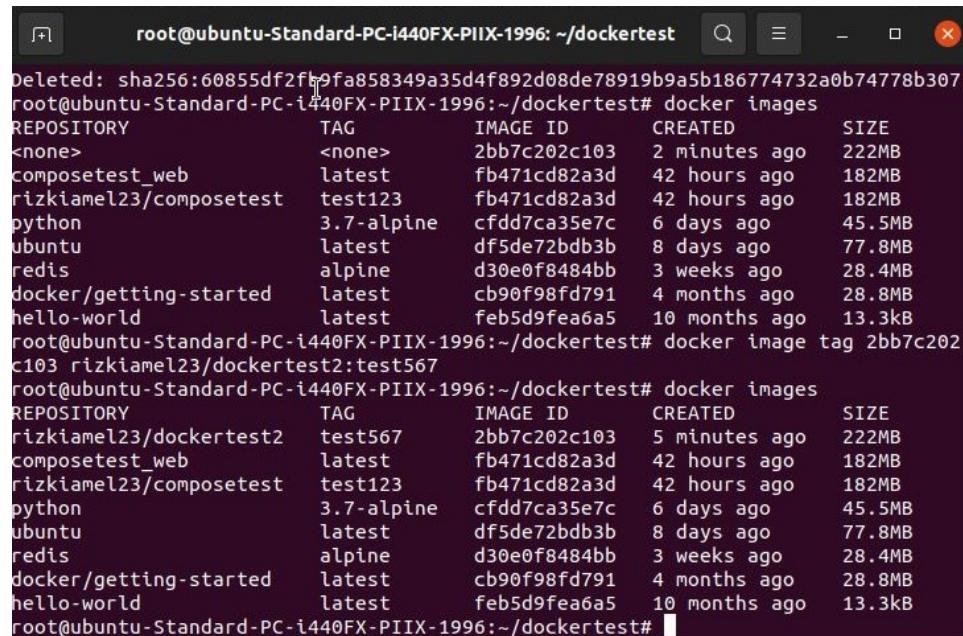
5. Beri nama dan tag pada docker image

Beri nama dan tag : docker image tag <IMAGE ID> <NAMA IMAGE:TAG>

COMMAND: `docker image tag 2bb7c202c103 rizkiamel23/dockertest2:test567`

Cek apakah nama dan tag docker images berhasil diganti :

COMMAND: `docker images`



```

Deleted: sha256:60855df2fb9fa858349a35d4f892d08de78919b9a5b186774732a0b74778b307
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker images
REPOSITORY          TAG      IMAGE ID      CREATED     SIZE
<none>            <none>   2bb7c202c103  2 minutes ago  222MB
composetest_web    latest   fb471cd82a3d  42 hours ago   182MB
rizkiamel23/composetest  test123  fb471cd82a3d  42 hours ago   182MB
python              3.7-alpine  cfdd7ca35e7c  6 days ago    45.5MB
ubuntu              latest   df5de72bdb3b  8 days ago    77.8MB
redis               alpine   d30e0f8484bb  3 weeks ago   28.4MB
docker/getting-started  latest   cb90f98fd791  4 months ago   28.8MB
hello-world         latest   feb5d9fea6a5  10 months ago  13.3kB
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker image tag 2bb7c202c103 rizkiamel23/dockertest2:test567
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker images
REPOSITORY          TAG      IMAGE ID      CREATED     SIZE
rizkiamel23/dockertest2  test567  2bb7c202c103  5 minutes ago  222MB
composetest_web    latest   fb471cd82a3d  42 hours ago   182MB
rizkiamel23/composetest  test123  fb471cd82a3d  42 hours ago   182MB
python              3.7-alpine  cfdd7ca35e7c  6 days ago    45.5MB
ubuntu              latest   df5de72bdb3b  8 days ago    77.8MB
redis               alpine   d30e0f8484bb  3 weeks ago   28.4MB
docker/getting-started  latest   cb90f98fd791  4 months ago   28.8MB
hello-world         latest   feb5d9fea6a5  10 months ago  13.3kB
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest#

```

Figure 3.5: Tag docker images

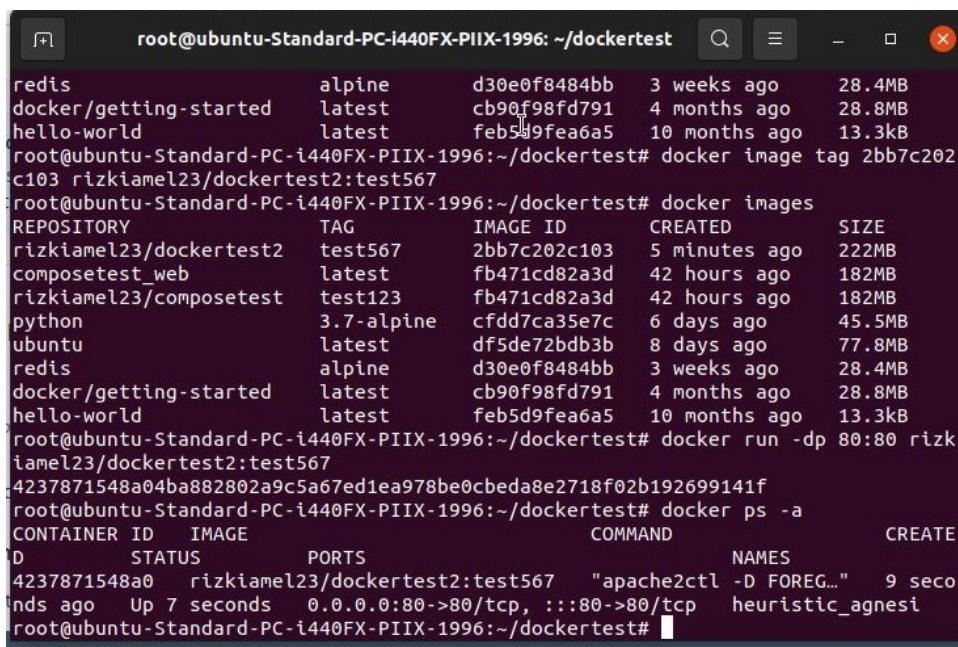
Keterangan : Jika ada rencana memasukan docker images pada repositori Docker / Docker Hub maka berikan nama sesuai repositori yang akan dibuat dengan format <USERNAME>/<NAMA REPOSITORI>

### 3.3 Run Pada Docker

- Buat dan run container dari docker images yang sudah dibuat

Buat dan run container dari docker images : docker run <OPTION> <NAMA IMAGE:TAG>

COMMAND: `docker run -dp 80:80 rizkiamel23/dockertest:test567`



```

root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~/dockertest
redis           alpine      d30e0f8484bb  3 weeks ago   28.4MB
docker/getting-started  latest      cb90f98fd791  4 months ago  28.8MB
hello-world     latest      feb5d9fea6a5  10 months ago 13.3kB
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker image tag 2bb7c202c103 rizkiamel23/dockertest2:test567
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
rizkiamel23/dockertest2  test567  2bb7c202c103  5 minutes ago  222MB
composetest_web     latest   fb471cd82a3d  42 hours ago  182MB
rizkiamel23/composetest  test123  fb471cd82a3d  42 hours ago  182MB
python              3.7-alpine  cfdd7ca35e7c  6 days ago   45.5MB
ubuntu              latest   df5de72bdb3b  8 days ago   77.8MB
redis               alpine      d30e0f8484bb  3 weeks ago   28.4MB
docker/getting-started  latest   cb90f98fd791  4 months ago  28.8MB
hello-world         latest   feb5d9fea6a5  10 months ago 13.3kB
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker run -dp 80:80 rizkiamel23/dockertest2:test567
4237871548a04ba882802a9c5a67ed1ea978be0cbeda8e2718f02b192699141f
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
4237871548a0        rizkiamel23/dockertest2:test567   "apache2ctl -D FOREG..."   9 seconds ago       Up 7 seconds        0.0.0.0:80->80/tcp, :::80->80/tcp   heuristic_agnesi
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest#

```

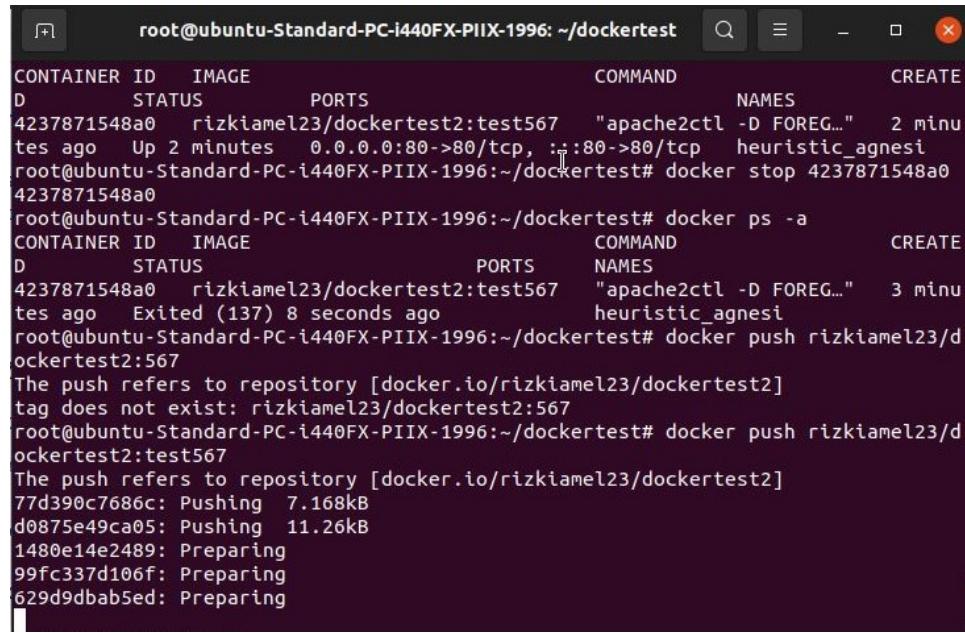
Figure 3.6: Buat dan run images

Keterangan :

- Opsi -d untuk mengatur ke detached mode / terpisah, agar container bisa berjalan pada background
- Opsi p untuk memetakan port 80 localhost yang akan mengakses port host 80 yang sudah di expose di file Dockerfile

- Cek apakah kontainer berjalan

COMMAND: `docker ps -a`



```

root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~/dockertest
CONTAINER ID IMAGE COMMAND NAMES
D STATUS PORTS NAMES
4237871548a0 rizkiamel23/dockertest2:test567 "apache2ctl -D FOREG..." 2 minutes ago Up 2 minutes 0.0.0.0:80->80/tcp, :::80->80/tcp heuristic_agnesi
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker stop 4237871548a0
4237871548a0
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker ps -a
CONTAINER ID IMAGE COMMAND NAMES
D STATUS PORTS NAMES
4237871548a0 rizkiamel23/dockertest2:test567 "apache2ctl -D FOREG..." 3 minutes ago Exited (137) 8 seconds ago heuristic_agnesi
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker push rizkiamel23/dockertest2:567
The push refers to repository [docker.io/rizkiamel23/dockertest2]
tag does not exist: rizkiamel23/dockertest2:567
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker push rizkiamel23/dockertest2:test567
The push refers to repository [docker.io/rizkiamel23/dockertest2]
77d390c7686c: Pushing 7.168kB
d0875e49ca05: Pushing 11.26kB
1480e14e2489: Preparing
99fc337d106f: Preparing
629d9dbab5ed: Preparing

```

Figure 3.7: Cek kontainer

Buka broser ketikan `http://0.0.0.0:80`

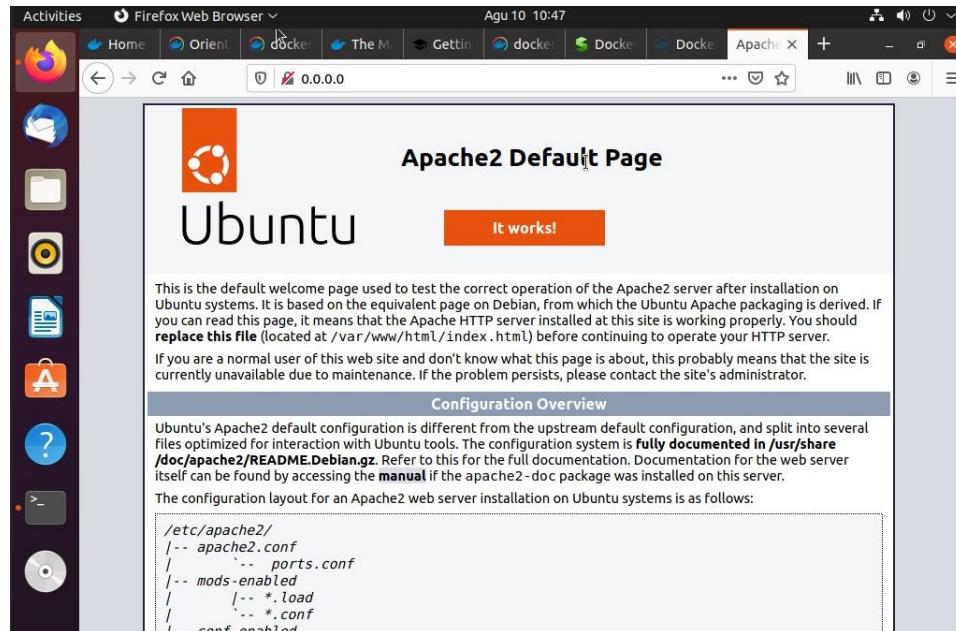


Figure 3.8: Tampilan akses ke kontainer

Apabila ingin menghentikan kontainer :  
**docker stop <CONTAINER ID>**  
 Apabila ingin menjalankan kembali kontainer : **docker start <CONTAINER ID>**

Apabila ingin menghapus kontainer : **docker rm <CONTAINER ID>**

4

## Upload dan Pull Pada Image

## 4.1 Pembuka

Panduan kali ini membahas upload dan pull docker images. Upload yang dilakukan yaitu menyimpan docker images yang sudah dibuat ke repositori Docker Hub. Push merupakan perintah docker untuk mengupload docker image ke repositori akun docker hub. Pull yang dilakukan yaitu mengambil docker images yang sudah ada di docker registri ke penyimpanan lokal untuk digunakan. Pull merupakan perintah docker untuk mengambil docker images yang tersedia di docker registry. Dengan ini pengguna tidak perlu membuat Dockerfile untuk membuat docker image, cukup download saja yang sudah tersedia.

## 4.2 Upload Image ke Docker Hub

1. Buat akun pada Docker Hub/apabila sudah ada lakukan login ke Docker Hub, kemudian buat repositori baru dengan klik tombol CREATE REPOSITORY

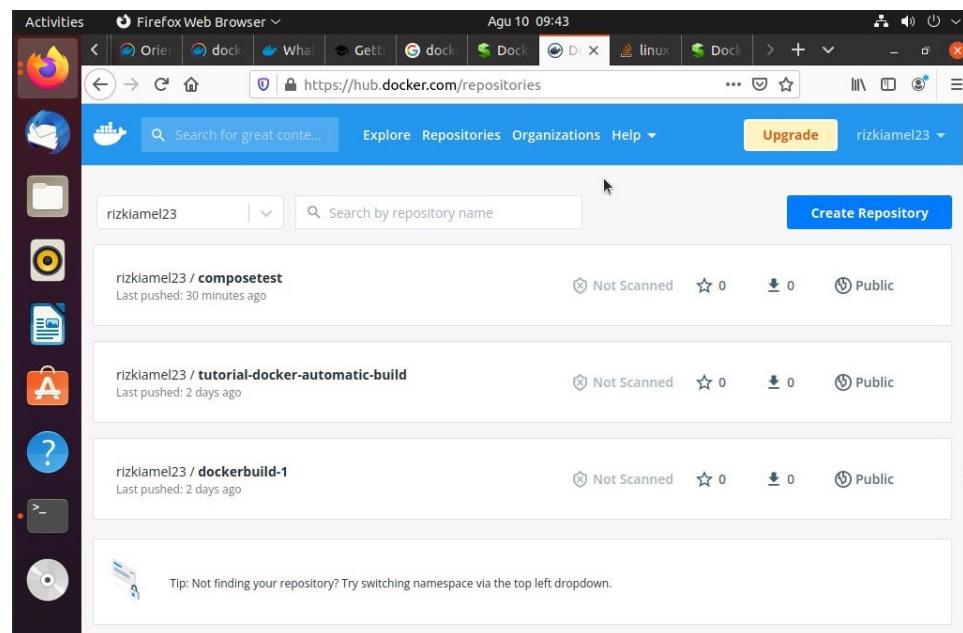


Figure 4.1: Buat repositori docker hub

2. Beri nama repositori sesuai dengan nama docker images yang sudah dibuat, tambahkan deskripsi(opsional), atur akses public/private, lalu klik tombol CREATE

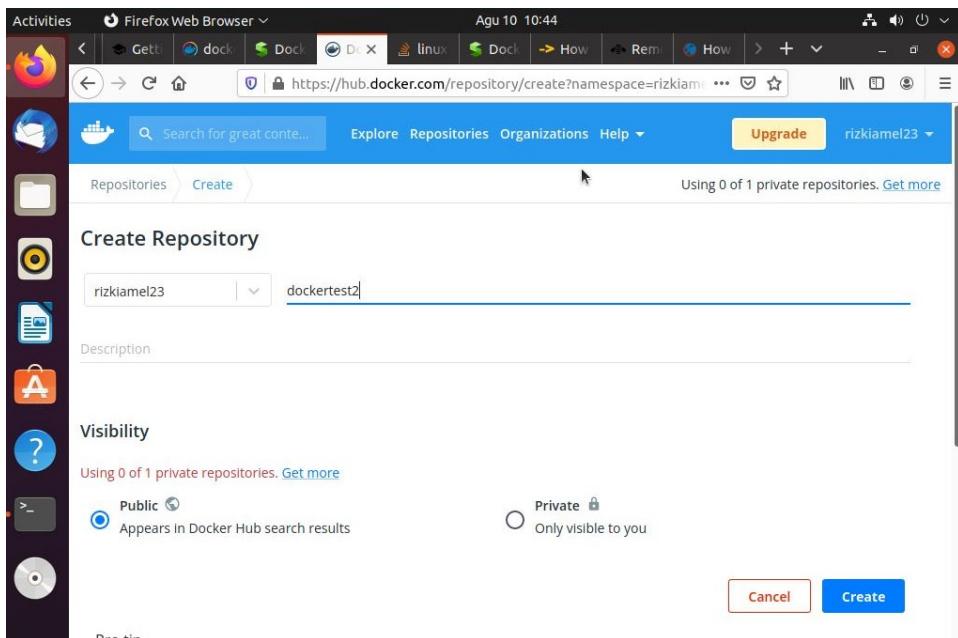


Figure 4.2: Pengaturan repositori

3. Jika berhasil maka tampilan terlihat seperti berikut

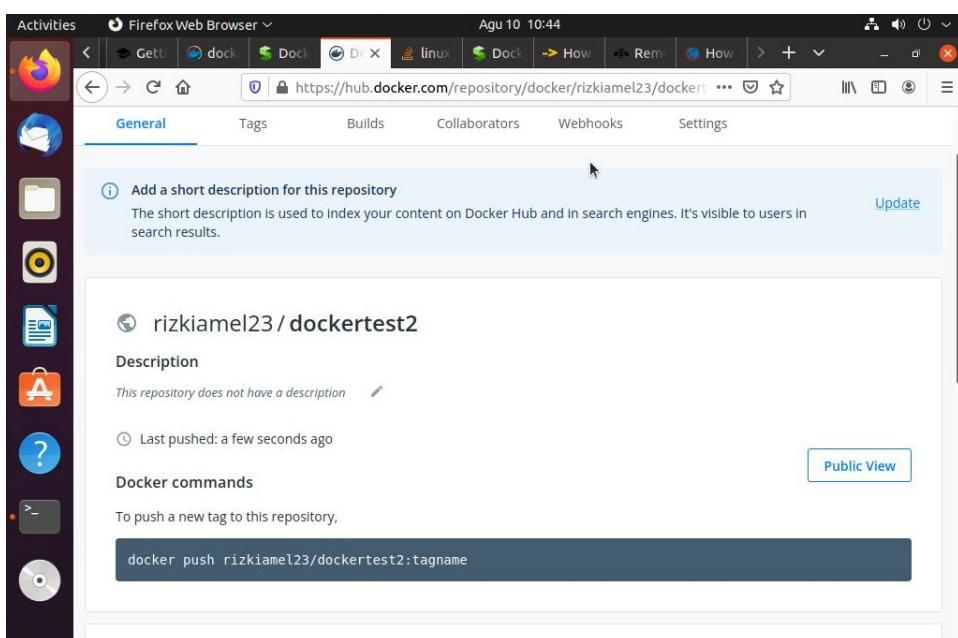
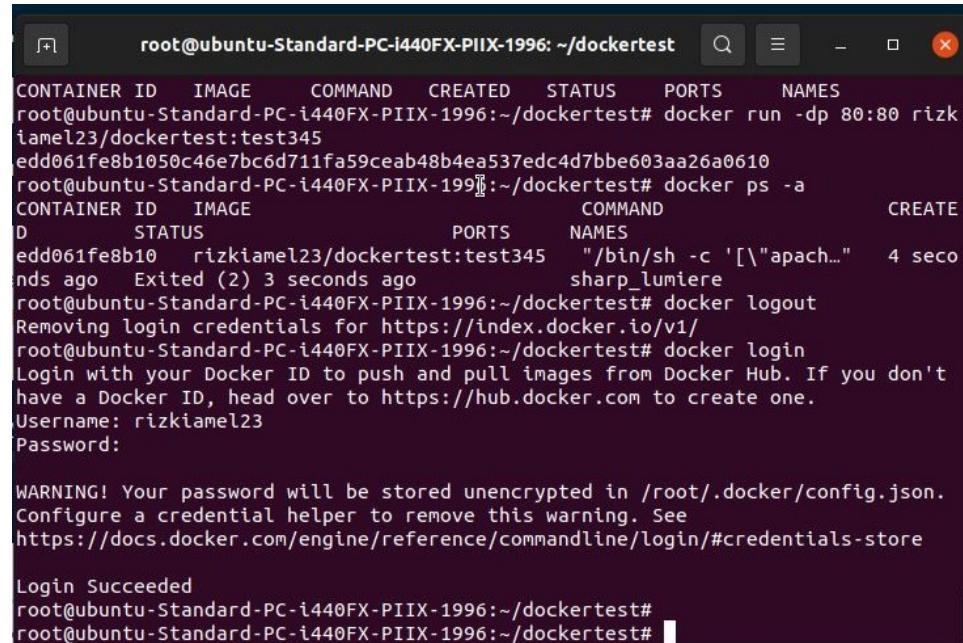


Figure 4.3: Berhasil membuat repositori

4. Lakukan login dengan akun docker di command line

COMMAND: [docker login](#), masukan USERNAME dan PASSWORD, klik ENTER, jika berhasil akan menampilkan login succeeded



```

root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~/dockertest
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker run -dp 80:80 rizkiamel23/dockertest:test345
edd061fe8b1050c46e7bc6d711fa59ceab48b4ea537edc4d7bbe603aa26a0610
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker ps -a
CONTAINER ID IMAGE COMMAND CREATE
D STATUS PORTS NAMES
edd061fe8b10 rizkiamel23/dockertest:test345 "/bin/sh -c '[\"apach...' 4 seconds ago Exited (2) 3 seconds ago sharp_lumiere
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker logout
Removing login credentials for https://index.docker.io/v1/
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: rizkiamel23
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest#
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest#

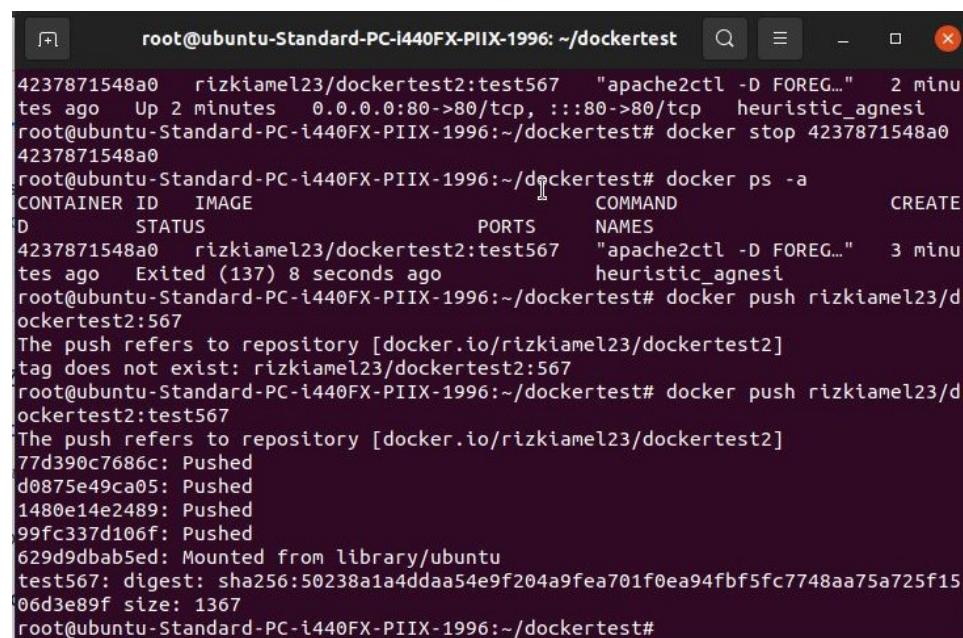
```

Figure 4.4: Login docker di command line

5. Lakukan upload docker image ke repositori docker hub yang sudah dibuat

Lakukan upload image : docker push <NAMA IMAGE:TAG>

COMMAND: [docker push rizkiamel23/dockertest2:test567](#)



```

root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~/dockertest
4237871548a0 rizkiamel23/dockertest2:test567 "apache2ctl -D FOREG..." 2 minutes ago Up 2 minutes 0.0.0.0:80->80/tcp, :::80->80/tcp heuristic_agnesi
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker stop 4237871548a0
4237871548a0
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker ps -a
CONTAINER ID IMAGE COMMAND CREATE
D STATUS PORTS NAMES
4237871548a0 rizkiamel23/dockertest2:test567 "apache2ctl -D FOREG..." 3 minutes ago Exited (137) 8 seconds ago heuristic_agnesi
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker push rizkiamel23/dockertest2:567
The push refers to repository [docker.io/rizkiamel23/dockertest2]
tag does not exist: rizkiamel23/dockertest2:567
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker push rizkiamel23/dockertest2:test567
The push refers to repository [docker.io/rizkiamel23/dockertest2]
77d390c7686c: Pushed
d0875e49ca05: Pushed
1480e14e2489: Pushed
99fc337d106f: Pushed
629d9dbab5ed: Mounted from library/ubuntu
test567: digest: sha256:50238a1a4ddaa54e9f204a9fea701f0ea94fbff5fc7748aa75a725f15
06d3e89f size: 1367
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest#

```

Figure 4.5: Upload docker images

6. Buka docker hub pada browser, jika berhasil pada tab tag and scans akan menampilkan tag dari image yang sudah diupload tadi

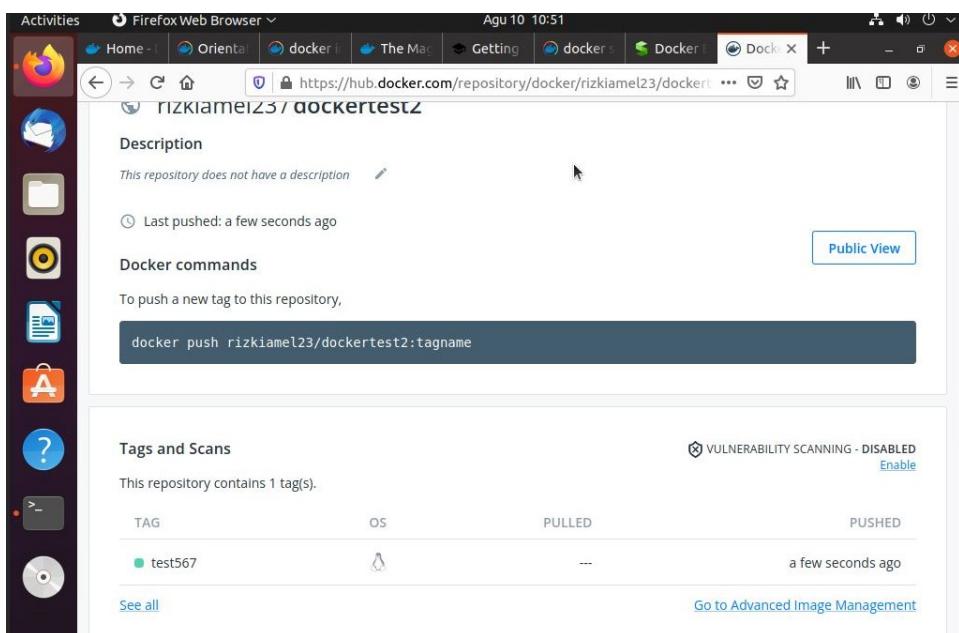


Figure 4.6: Tampilan berhasil upload

### 4.3 Pull Image dari Docker Registry

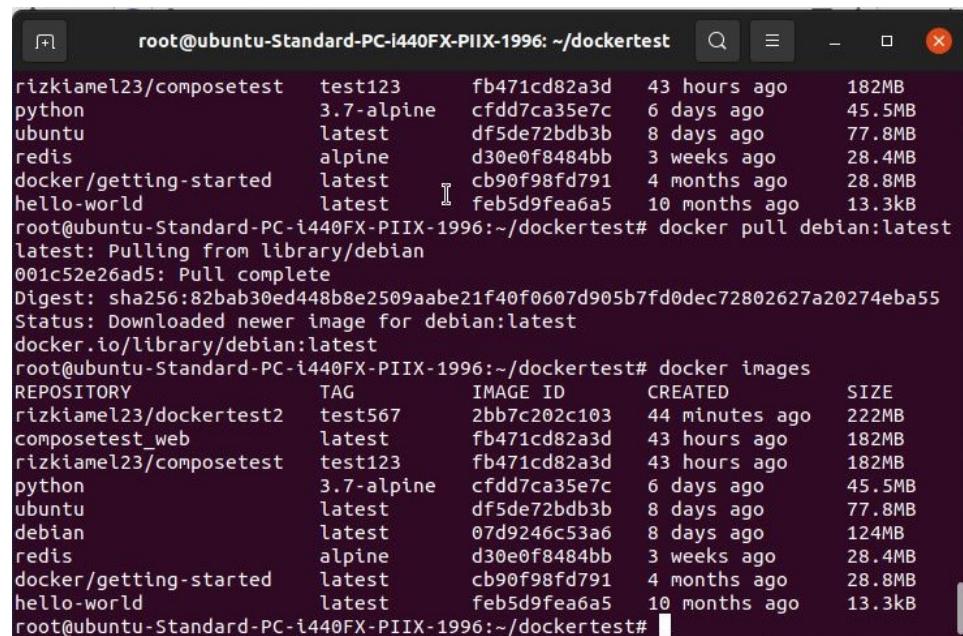
1. Pull image debian dari docker repositori

Pull image debian : docker pull <IMAGE NAME:TAG>

COMMAND: docker pull debian:latest

Cek docker images setelahnya

COMMAND: docker images



```

root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~/dockertest
rizkiamel23/composetest    test123      fb471cd82a3d  43 hours ago   182MB
python                      3.7-alpine   cfdd7ca35e7c  6 days ago    45.5MB
ubuntu                      latest       df5de72bdb3b  8 days ago    77.8MB
redis                       alpine      d30e0f8484bb  3 weeks ago   28.4MB
docker/getting-started     latest       cb90f98fd791  4 months ago  28.8MB
hello-world                 latest      feb5d9fea6a5  10 months ago 13.3kB
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker pull debian:latest
latest: Pulling from library/debian
001c52e26ads: Pull complete
Digest: sha256:82bab30ed448b8e2509aabeb21f40f0607d905b7fd0dec72802627a20274eba55
Status: Downloaded newer image for debian:latest
docker.io/library/debian:latest
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest# docker images
REPOSITORY          TAG      IMAGE ID   CREATED        SIZE
rizkiamel23/dockertest2  test567   2bb7c202c103  44 minutes ago  222MB
composetest_web     latest    fb471cd82a3d  43 hours ago   182MB
rizkiamel23/composetest  test123   fb471cd82a3d  43 hours ago   182MB
python              3.7-alpine cfdd7ca35e7c  6 days ago    45.5MB
ubuntu              latest    df5de72bdb3b  8 days ago    77.8MB
debian              latest    07d9246c53a6  8 days ago    124MB
redis               alpine    d30e0f8484bb  3 weeks ago   28.4MB
docker/getting-started  latest    cb90f98fd791  4 months ago  28.8MB
hello-world         latest    feb5d9fea6a5  10 months ago 13.3kB
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockertest#

```

Figure 4.7: Pull docker images

Setelah docker image di pull/diambil, pengguna dapat membangun kontainer seperti biasa dari docker image yang telah di pull dari registri.

# 5

## Docker Volume

### 5.1 Pembuka

Dikutip dari situs Docker Docs, Docker volume merupakan salah satu cara untuk memanage data di docker selain menggunakan Bind Mount. Dengan volume pengguna dapat menyimpan file pada host machine yang dapat diterapkan ke container, sehingga file tetap tersedia meskipun container dihapus. Volume disimpan di file host system yang dikelola oleh docker (/var/lib/docker/volumes). Saat membuat docker volume, folder ini akan menyimpan data docker volume. Saat mount volume ke dalam container, direktori ini yang di mount ke dalam container. Docker menyediakan 2 opsi sintaks untuk membuat volume yaitu -v/-volume dan -mount. Pada kesempatan kali ini akan menggunakan opsi -v/-volume

### 5.2 Kapan menggunakan volume ?

- Berbagi data diantara beberapa container yang sedang berjalan
- Menyimpan data container di remote host atau provider
- Perlu mencadangkan (backup), memulihkan (restore), atau memigrasikan (migrate) data dari satu docker host ke docker host lain
- Aplikasi membutuhkan I/O berkinerja di Docker Desktop
- Aplikasi memerlukan perilaku native filesystem di Docker Desktop

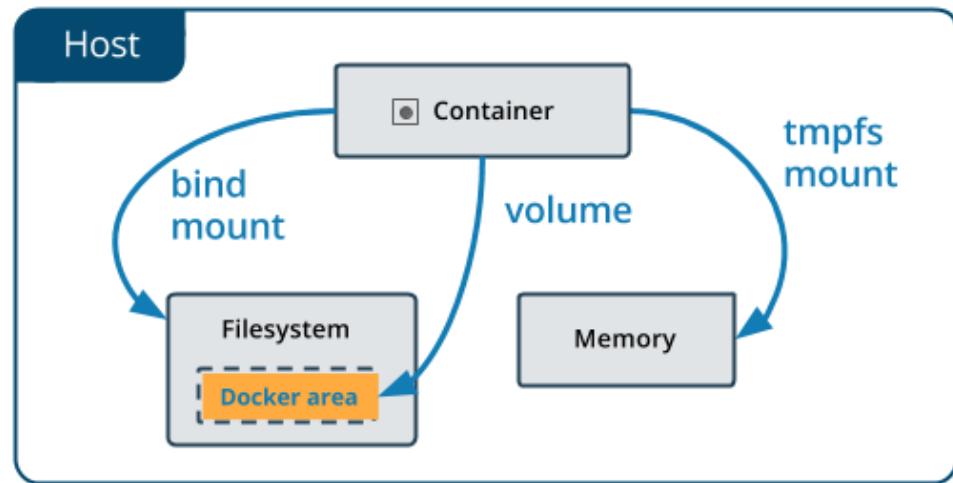


Figure 5.1: Docker Volume

### 5.3 Membuat docker Volume

1. Membuat docker volume saat container dibuat

Membuat docker volume saat container dibuat : docker run <OPTION> -v <NAMA VOLUME: MOUNT DIREKTORI CONTAINER> -name <NAMA CONTAINER> <NAMA IMAGE>

COMMAND: docker run -dp 80:80 -v volume1:/usr/share/nginx/html -name web nginx:stable-alpine

Jika sudah cek Volume apakah sudah terbuat

COMMAND: docker volume ls

Lalu cek container apakah berjalan

COMMAND: docker ps -a

```

root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~
root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~# docker run -dp 80:80 -v volume1:/usr/share/nginx/html --name web nginx:stable-alpine
19c36e4e9c886613331ae762f5499f46120032e3e39047f128686e7a44cb2d48
root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~# docker volume ls
DRIVER      VOLUME NAME
local      43ca42678aa99d34d5d7bc72e9eed95e844cb4874cd4e66dd9febb2937365f84
local      volume1
root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~# docker ps -a
CONTAINER ID   IMAGE          COMMAND           CREATED          NAMES
STATUS          PORTS          NAMES
19c36e4e9c88   nginx:stable-alpine    "/docker-entrypoint..."   About a minute ago
Up About a minute  0.0.0.0:80->80/tcp, :::80->80/tcp   web
4237871548a0   rizkiamel23/dockertest2:test567  "apache2ctl -D FOREG..."   2 hours ago
Exited (137) 2 hours ago                                     heuristic_agnesi
root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~#

```

Figure 5.2: Buat docker volume saat container dibuat

2. Buka browser akses `http://0.0.0.0:80`, akan muncul tampilan berikut

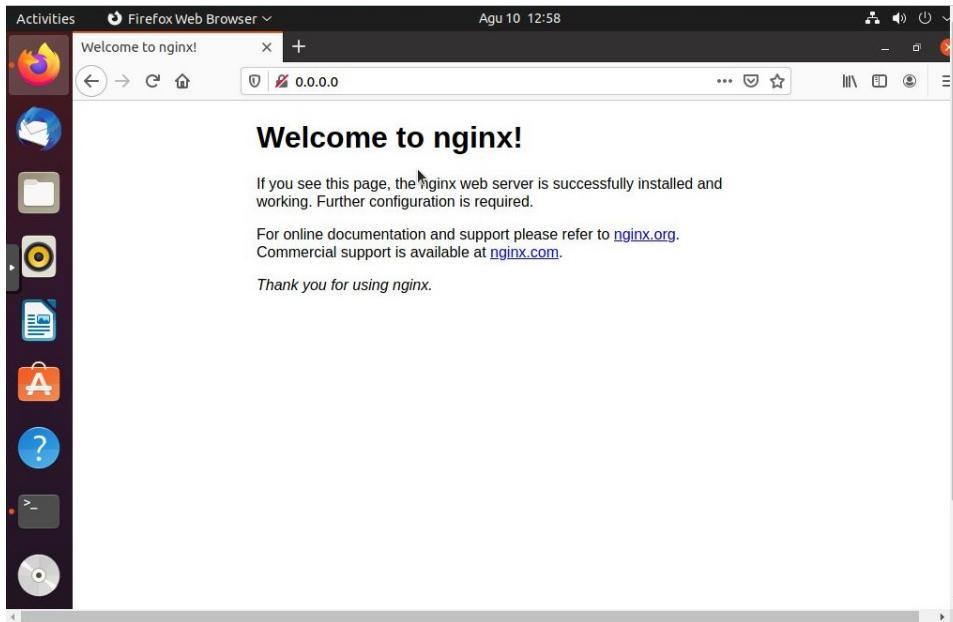


Figure 5.3: Tampilan mengakses kontainer dalam volume

3. Periksa detail docker volume

Periksa detail docker volume : docker volume inspect <NAMA VOLUME>  
 COMMAND: `docker volume inspect volume1`

```

root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# docker volume ls
DRIVER      VOLUME NAME
local        43ca42678aa99d34d5d7bc72e1bed95e844cb4874cd4e66dd9febb2937365f84
local        volume1
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED             NAMES
STATUS          PORTS          NAMES
19c36e49c88   nginx:stable-alpine    "/docker-entrypoint..."   About a minute ago
Up About a minute   0.0.0.0:80->80/tcp, :::80->80/tcp   web
4237871548a0   rizkiamel23/dockertest2:test567  "apache2ctl -D FORE..."   2 hours ago
Exited (137) 2 hours ago           heuristic_agnesi
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# docker volume inspect volume1
[{"Created": "2022-08-10T12:56:15+07:00", "Driver": "local", "Labels": null, "Mountpoint": "/var/lib/docker/volumes/volume1/_data", "Name": "volume1", "Options": null, "Scope": "local"}]
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~#

```

Figure 5.4: Tampilan inspect docker volume



# 6

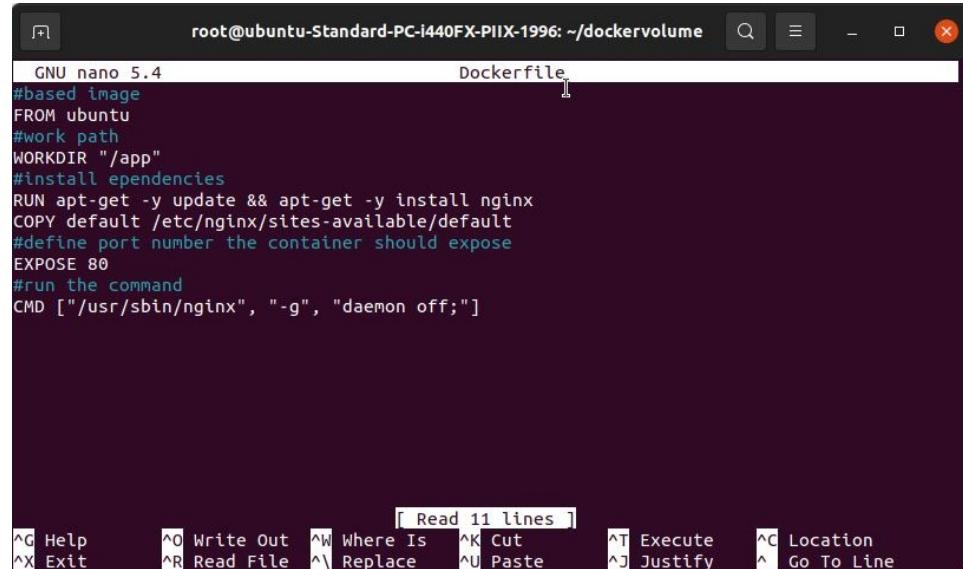
## Memanipulasi File Pada Docker volume

### 6.1 Pembuka

Panduan kali ini akan membahas mengenai manipulasi file pada docker volume. Pertama akan membuat dockerfile dengan tambahan konfigurasi WORKDIR. WORKDIR merupakan sintaks untuk mengarahkan arah direktori kerja /app volume yang akan di mount ke container nanti. File akan dibuat pada direktori ini, Kemudian user lain akan mengedit, melihat file yang disimpan pada direktori ini.

### 6.2 Langkah memanipulasi file pada docker volume

- Buat dockerfile tambahkan data berikut dan baris WORKDIR "/app"



```

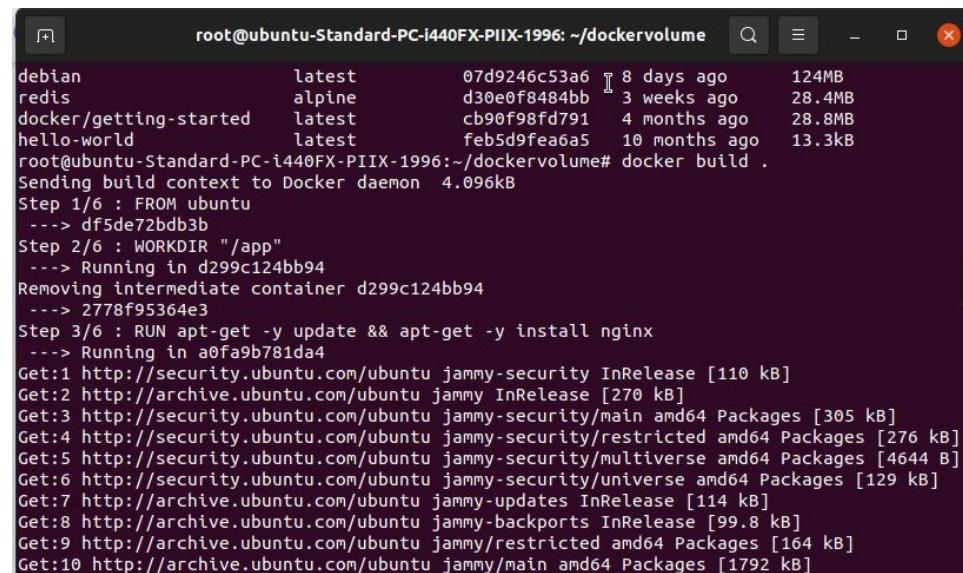
GNU nano 5.4                               Dockerfile
#based image
FROM ubuntu
#work path
WORKDIR "/app"
#install ependencies
RUN apt-get -y update && apt-get -y install nginx
COPY default /etc/nginx/sites-available/default
#define port number the container should expose
EXPOSE 80
#run the command
CMD ["/usr/sbin/nginx", "-g", "daemon off;"]

```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location  
 ^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^\_ Go To Line

Figure 6.1: Buat Dockerfile

- Buat docker images dari Dockerfile yang dibuat tadi  
 COMMAND: docker build .



```

root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~/dockervolume#
root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~/dockervolume# docker build .
Sending build context to Docker daemon 4.096kB
Step 1/6 : FROM ubuntu
--> df5de72bdb3b
Step 2/6 : WORKDIR "/app"
--> Running in d299c124bb94
Removing intermediate container d299c124bb94
--> 2778f95364e3
Step 3/6 : RUN apt-get -y update && apt-get -y install nginx
--> Running in a0fa9b781da4
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [305 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [276 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [4644 B]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [129 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy/restricted amd64 Packages [164 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1792 kB]

```

Figure 6.2: Buat docker images

## 6.2. LANGKAH MEMANIPULASI FILE PADA DOCKER VOLUME 29

3. Beri nama dan tag pada image

COMMAND: `docker image tag 006ff0a7b2ea rizkiamel23/web:test678`

Lihat docker images

COMMAND: `docker images`

The terminal window shows the root user on an Ubuntu system. It lists several Docker images with their details (REPOSITORY, TAG, IMAGE ID, CREATED, SIZE). The user runs the command `docker image tag 006ff0a7b2ea rizkiamel23/web:test678`. After this, the user runs `docker images` again, which shows the updated list including the newly tagged image.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
rizkiamel23/web	test678	006ff0a7b2ea	3 minutes ago	169MB
rizkiamel23/dockertest2	test567	2bb7c202c103	3 hours ago	222MB
nginx	stable-alpine	075b447b534f	10 hours ago	23.5MB
composetest_web	latest	fb471cd82a3d	46 hours ago	182MB
rizkiamel23/composetest	test123	fb471cd82a3d	46 hours ago	182MB
python	3.7-alpine	cfdd7ca35e7c	6 days ago	45.5MB
python	3.8	4544558e07c5	6 days ago	913MB
ubuntu	latest	df5de72bdb3b	8 days ago	77.8MB
debian	latest	07d9246c53a6	8 days ago	124MB
redis	alpine	d30e0f8484bb	3 weeks ago	28.4MB
docker/getting-started	latest	cb90f98fd791	4 months ago	28.8MB
hello-world	latest	feb5d9fea6a5	10 months ago	13.3kB

Figure 6.3: Beri nama dan tag images

4. Buat volume dengan container, masuk direktori container dan buat file baru

Buat volume dengan container: `docker run <OPTION> -name <NAMA CONTAINER> -v <NAMA VOLUME:MOUNT DIREKTORI CONTAINER> <NAMA IMAGE:TAG> bash`

COMMAND: `docker run -v volume5:/app rizkiamel23/web:text678 bash`

Setelah masuk bash, buat file baru

COMMAND: `cat > index.html`, isi file, Klik CTRL+D

The terminal window shows the root user on an Ubuntu system. The user creates a volume named `volume4`. Then, they run a Docker container with the volume mounted at `/app`. Inside the container, the user creates a file named `index.html` and writes "Hello World !". Finally, they exit the container and list the files in the `/app` directory.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
rizkiamel23/web	test678	006ff0a7b2ea	2 days ago	169MB
rizkiamel23/dockertest2	test567	2bb7c202c103	2 days ago	222MB
nginx	stable-alpine	075b447b534f	2 days ago	23.5MB
composetest_web	latest	fb471cd82a3d	3 days ago	182MB
rizkiamel23/composetest	test123	fb471cd82a3d	3 days ago	182MB
python	3.7-alpine	cfdd7ca35e7c	8 days ago	45.5MB
python	3.8	4544558e07c5	8 days ago	913MB
ubuntu	latest	df5de72bdb3b	10 days ago	77.8MB
debian	latest	07d9246c53a6	10 days ago	124MB
redis	alpine	d30e0f8484bb	3 weeks ago	28.4MB
docker/getting-started	latest	cb90f98fd791	4 months ago	28.8MB
hello-world	latest	feb5d9fea6a5	10 months ago	13.3kB

Figure 6.4: Buat volume, masuk direktori container

### 5. Cek file mount

Buka terminal baru :

Klik kanan terminal, new windows

Masuk akun root :

COMMAND: `sudo -i`, masukan password root

Cek docker volume :

COMMAND: `docker ps -a`

Masuk ke direktori container : `docker exec -t -i <ID CONTAINER> /bin/bash`

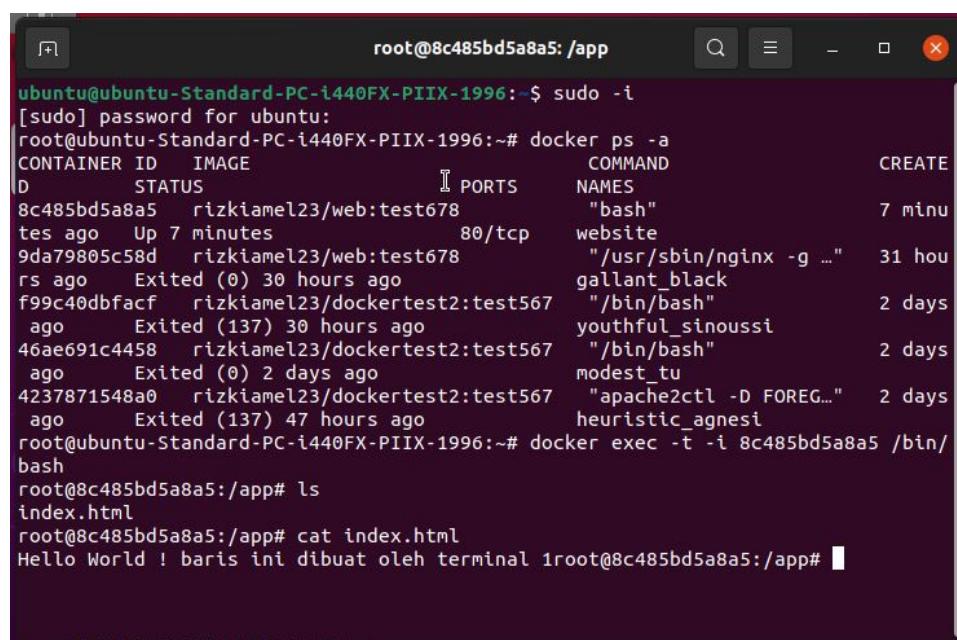
COMMAND: `docker exec -t -i 8c485bd5a8a5 /bin/bash`

Lihat list file :

COMMAND: `ls`

Lihat isi file index.html :

COMMAND: `cat index.html`



```

root@8c485bd5a8a5:/app# sudo -i
[sudo] password for ubuntu:
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# docker ps -a
CONTAINER ID   IMAGE          COMMAND
D              STATUS          PORTS     NAMES
8c485bd5a8a5   rizkiamel23/web:test678   "bash"
7 minu
tes ago        Up 7 minutes   80/tcp    website
9da79805c58d   rizkiamel23/web:test678   "/usr/sbin/nginx -g ..."
31 hou
rs ago         Exited (0) 30 hours ago   gallant_black
f99c40dbfacf   rizkiamel23/dockertest2:test567   "/bin/bash"
2 days
ago           Exited (137) 30 hours ago   youthful_sinoussi
46ae691c4458   rizkiamel23/dockertest2:test567   "/bin/bash"
2 days
ago           Exited (0) 2 days ago      modest_tu
4237871548a0   rizkiamel23/dockertest2:test567   "apache2ctl -D FOREG..."
2 days
ago           Exited (137) 47 hours ago   heuristic_agnesi
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# docker exec -t -i 8c485bd5a8a5 /bin/
bash
root@8c485bd5a8a5:/app# ls
index.html
root@8c485bd5a8a5:/app# cat index.html
Hello World ! baris ini dibuat oleh terminal 1root@8c485bd5a8a5:/app# 

```

Figure 6.5: Cek file mount

6. Lakukan inspect volume

Keluar dari mode bash :

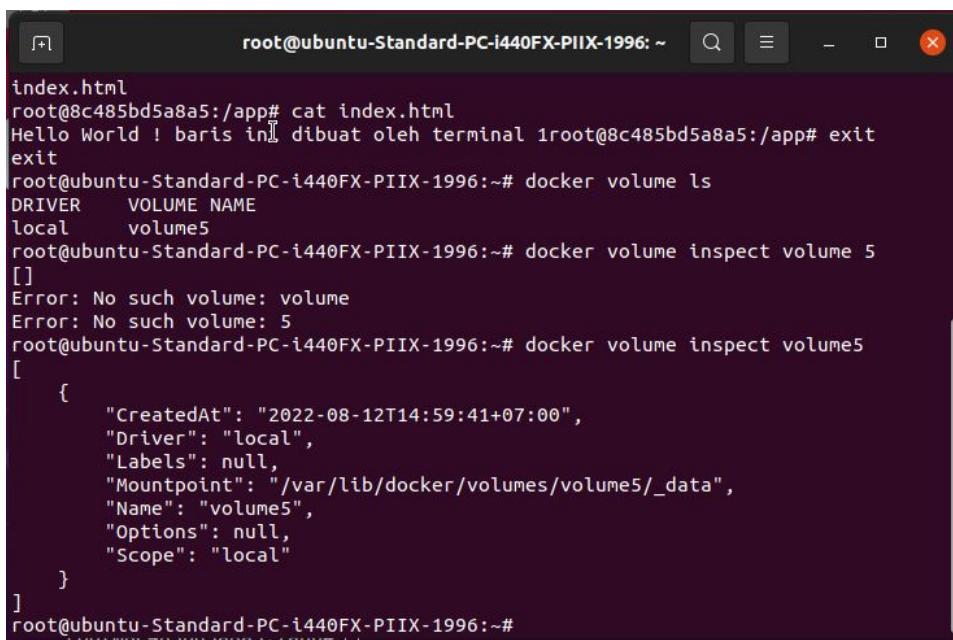
COMMAND: `exit`

Lihat list volume :

COMMAND: `docker volume ls`

Inspect volume : `docker volume inspect <NAMA VOLUME>`

COMMAND: `docker volume inspect volume5`



The screenshot shows a terminal window titled "root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~". The terminal output is as follows:

```
index.html
root@8c485bd5a8a5:/app# cat index.html
Hello World ! baris ini dibuat oleh terminal 1
root@8c485bd5a8a5:/app# exit
exit
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# docker volume ls
DRIVER      VOLUME NAME
local        volume5
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# docker volume inspect volume5
[]
Error: No such volume: volume5
Error: No such volume: 5
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# docker volume inspect volume5
[
    {
        "CreatedAt": "2022-08-12T14:59:41+07:00",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/volume5/_data",
        "Name": "volume5",
        "Options": null,
        "Scope": "local"
    }
]
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~#
```

Figure 6.6: Inspect volume

Dari gambar diatas dapat dilihat path direktori yang di mounted oleh volume, path ini merupakan tempat penyimpanan pada volume. walaupun container dihapus selama volumnya tidak dihapus direktori ini tetap ada, direktori ini dapat dihapus jika volumnya dihapus.

### 7. Masuk direktori mount volume

Masuk direktori mount volume : cd <PATH DIREKTORI MOUNT>

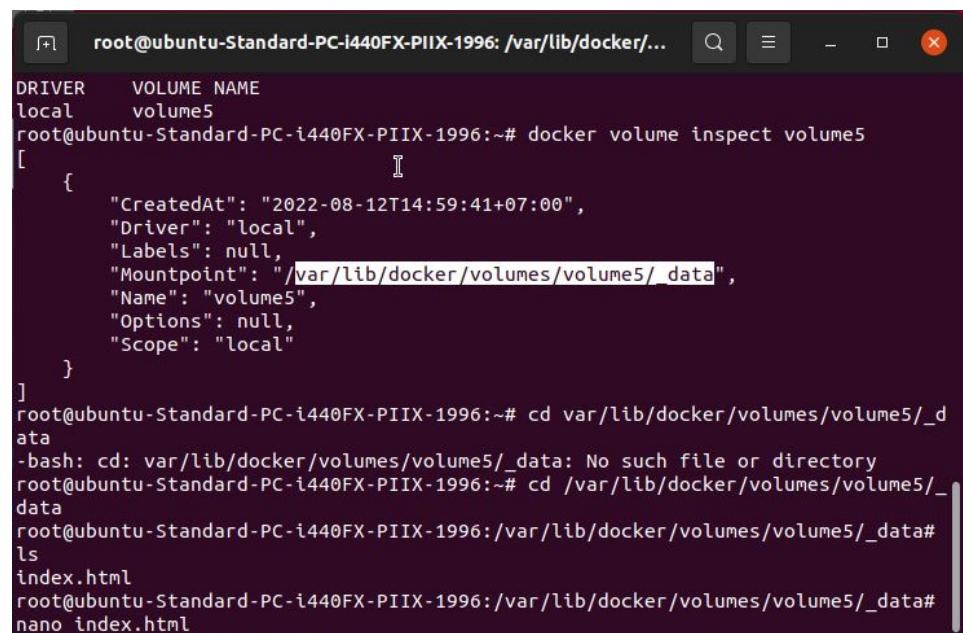
COMMAND: cd /var/lib/docker/volumes/volume5/\_data

Lihat list file :

COMMAND: ls

Edit file :

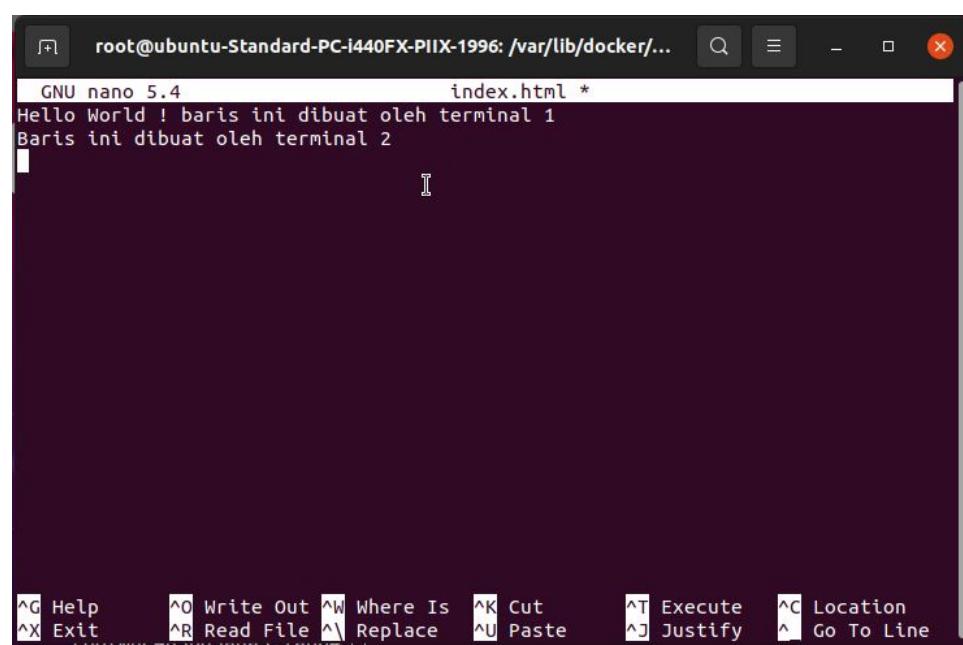
COMMAND: nano index.html



```
root@ubuntu-Standard-PC-i440FX-PIIX-1996: /var/lib/docker/...
DRIVER      VOLUME NAME
local        volume5
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# docker volume inspect volume5
[{"Driver": "local", "Name": "volume5", "Mountpoint": "/var/lib/docker/volumes/volume5/_data", "Options": {}, "Scope": "local", "Labels": null, "CreatedAt": "2022-08-12T14:59:41+07:00", "Driver": "local", "Mountpoint": "/var/lib/docker/volumes/volume5/_data", "Name": "volume5", "Options": null, "Scope": "local"}]
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# cd var/lib/docker/volumes/volume5/_data
-bash: cd: var/lib/docker/volumes/volume5/_data: No such file or directory
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# cd /var/lib/docker/volumes/volume5/_data
root@ubuntu-Standard-PC-i440FX-PIIX-1996:/var/lib/docker/volumes/volume5/_data# ls
index.html
root@ubuntu-Standard-PC-i440FX-PIIX-1996:/var/lib/docker/volumes/volume5/_data# nano index.html
```

Figure 6.7: Edit file

Kemudian lakukan edit file lalu tekan CTRL+X, Y, ENTER



```
root@ubuntu-Standard-PC-i440FX-PIIX-1996: /var/lib/docker/...
GNU nano 5.4          index.html *
Hello World ! baris ini dibuat oleh terminal 1
Baris ini dibuat oleh terminal 2
```

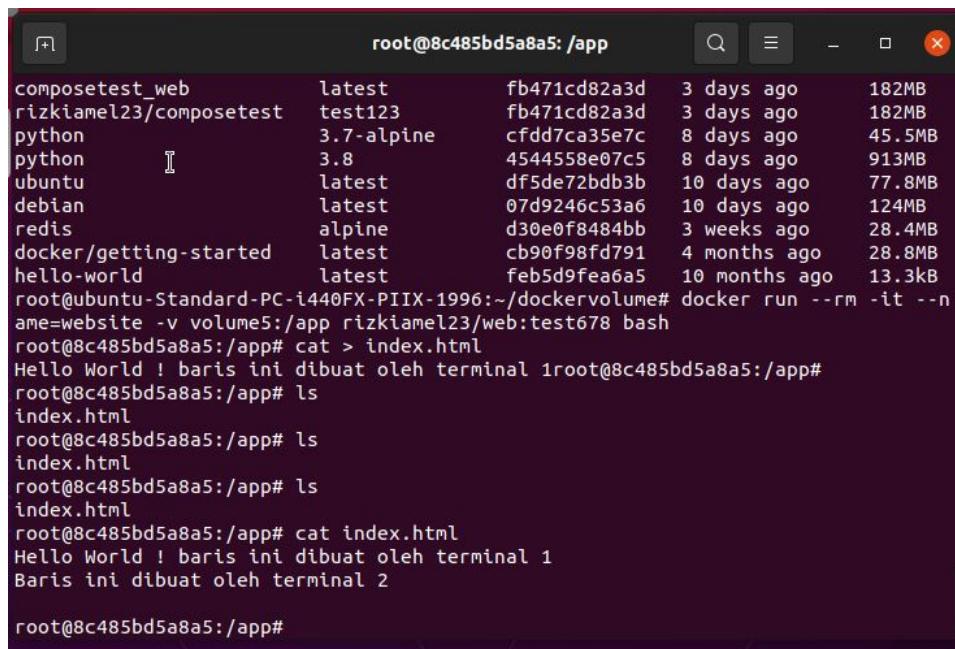
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location  
 ^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^^ Go To Line

Figure 6.8: Edit file dan simpan

8. Buka terminal sebelumnya, cek isi file

Cek isi file :

COMMAND: cat index.html



The screenshot shows a terminal window with the following content:

```
root@8c485bd5a8a5: /app
latest      fb471cd82a3d  3 days ago  182MB
rizkiamel23/composetest test123    fb471cd82a3d  3 days ago  182MB
python       3.7-alpine   cfd7ca35e7c  8 days ago  45.5MB
python       3.8         4544558e07c5  8 days ago  913MB
ubuntu       latest      df5de72bdb3b  10 days ago 77.8MB
debian       latest      07d9246c53a6  10 days ago 124MB
redis        alpine     d30e0f8484bb  3 weeks ago 28.4MB
docker/getting-started latest      cb90f98fd791  4 months ago 28.8MB
hello-world  latest      feb5d9fea6a5  10 months ago 13.3kB
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockervolume# docker run --rm -it --name=website -v volume5:/app rizkiamel23/web:test678 bash
root@8c485bd5a8a5:/app# cat > index.html
Hello World ! baris ini dibuat oleh terminal 1
root@8c485bd5a8a5:/app# ls
index.html
root@8c485bd5a8a5:/app# ls
index.html
root@8c485bd5a8a5:/app# cat index.html
Hello World ! baris ini dibuat oleh terminal 1
Baris ini dibuat oleh terminal 2
root@8c485bd5a8a5:/app#
```

Figure 6.9: Cek isi file

Dari gambar diatas file bisa dimodifikasi selama volume belum dihapus



# 7

# Docker Network

## 7.1 Pengertian

Dikutip dari situs Docker Docs, Docker network merupakan sistem jaringan docker untuk menghubungkan antar container docker atau menghubungkannya ke beban kerja non-docker. Kontainer dan layanan docker bahkan tidak perlu menyadari bahwa mereka di deploy di docker, atau apakah container/layanan docker lain merupakan beban kerja docker atau tidak. Baik host docker pengguna menjalankan linux, windows atau campuran keduanya, pengguna dapat menggunakan docker untuk mengelolanya dengan cara diagnostik platform.

## 7.2 Docker Network Drivers

Subsistem docker network dapat dihubungkan menggunakan driver. Beberapa driver ada secara default dan menyediakan fungsionalitas jaringan inti diantaranya :

### 7.2.1 Bridge (default)

merupakan konfigurasi default server docker network jika pengguna tidak mendefinisikan driver saat membuat docker network. Bridge biasanya digunakan saat aplikasi yang berjalan dalam kontainer secara mandiri perlu berkomunikasi. Pengguna dapat menggunakan driver ini untuk komunikasi antara kontainer dalam docker host yang sama.

### 7.2.2 Host

merupakan konfigurasi untuk kontainer mandiri, menghapus isolasi jaringan antara kontainer dan docker host, dan menggunakan jaringan host secara langsung. Pengguna dapat menggunakan driver ini ketika tumpukan jaringan

tidak boleh diisolasi dari docker host tapi mengiginkan aspek lain dari kontainer untuk diisolasi.

### 7.2.3 Overlay

merupakan konfigurasi untuk menghubungkan beberapa docker daemon bersama sama dan memungkinkan layanan swarm untuk berkomunikasi satu sama lain. Strategi ini menghilangkan kebutuhan untuk melakukan perutean tingkat OS diantara kontainer. Pengguna dapat menggunakan driver ini untuk komunikasi antara kontainer dalam host docker yang berbeda atau saat beberapa aplikasi bekerja sama menggunakan layanan swarm.

### 7.2.4 IPvlan

merupakan konfigurasi yang memberi pengguna kendali penuh atas pen-galamatan IPv4 dan IPv6. Driver VLAN dibangun diatas itu dan memberi pengguna kendali penuh pada lapisan 2 VLAN tagging bahkan IPvlan L3 untuk pengguna yang tertarik pada integrasi jaringan underlay.

### 7.2.5 Macvlan

merupakan konfigurasi yang memperbolehkan pengguna untuk memasukan MAC Address ke kontainer, dan membuatnya tampak sebagai perangkat fisik di jaringan. Daemon docker merutekan lalu lintas ke kontainer dengan MAC Address. Pengguna dapat menggunakan driver ini agar kontainer terlihat seperti host fisik di jaringan, masing masing dengan MAC Address yang unik.

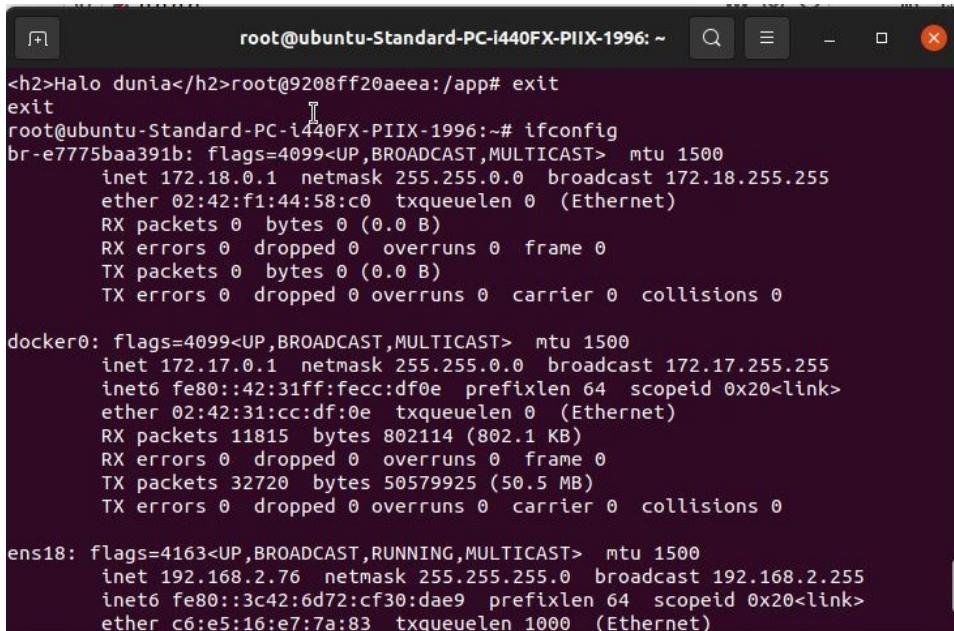
### 7.2.6 None

merupakan konfigurasi untuk menonaktifkan semua jaringan , biasanya digunakan bersama dengan driver jaringan khusus.

### 7.2.7 Network Plugins

merupakan konfigurasi untuk menginstal dan menggunakan plugin jaringan pihak ketiga dengan Docker. Plugin ini tersedia dari docker hub atau vendor pihak ketiga.

Pengguna dapat melihat port jaringan khusus untuk docker dalam linux  
COMMAND: `ifconfig`



```
<h2>Halo dunia</h2>root@9208ff20aeea:/app# exit
exit
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# ifconfig
br-e7775baa391b: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.18.0.1 netmask 255.255.0.0 broadcast 172.18.255.255
        ether 02:42:f1:44:58:c0 txqueuelen 0 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
        inet6 fe80::42:31ff:fecc:df0e prefixlen 64 scopeid 0x20<link>
        ether 02:42:31:cc:df:0e txqueuelen 0 (Ethernet)
        RX packets 11815 bytes 802114 (802.1 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 32720 bytes 50579925 (50.5 MB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens18: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.2.76 netmask 255.255.255.0 broadcast 192.168.2.255
        inet6 fe80::3c42:6d72:cf30:dae9 prefixlen 64 scopeid 0x20<link>
        ether c6:e5:16:e7:7a:83 txqueuelen 1000 (Ethernet)
```

Figure 7.1: Lihat port jaringan



# 8

## Menghubungkan Lebih Dari 1 Aplikasi Dengan Docker Network

### 8.1 Pembuka

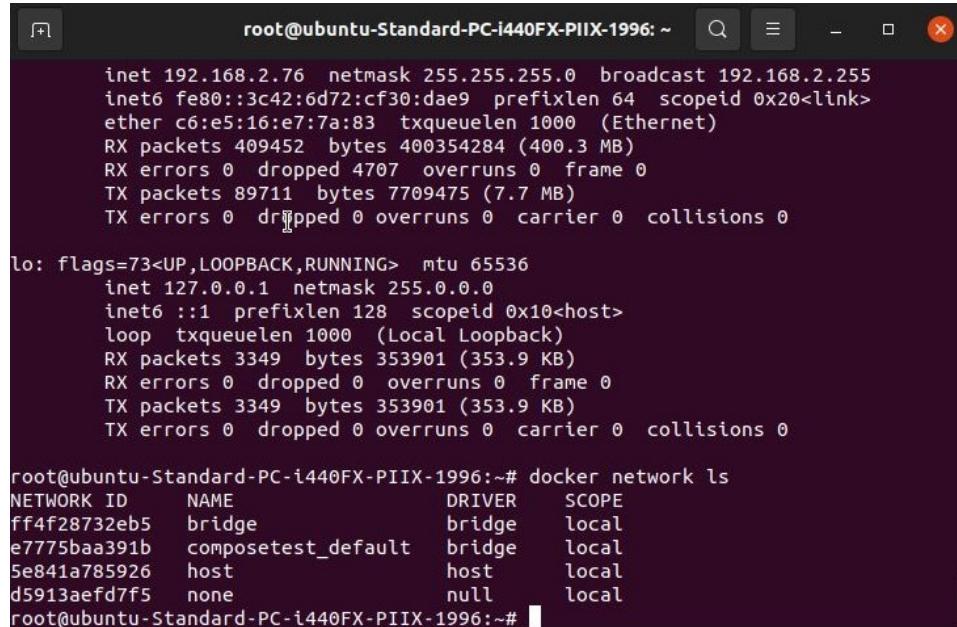
Pada panduan kali ini akan membahas mengenai menghubungkan aplikasi menggunakan docker network. Driver docker network yang digunakan adalah bridge. Kontainer yang dibuat akan dihubungkan dengan docker netwrok agar dapat terhubung.

### 8.2 Langkah menghubungkan lebih dari 1 aplikasi dengan docker network

## 40 8. MENGHUBUNGKAN LEBIH DARI 1 APLIKASI DENGAN DOCKER NETWORK

1. Cek docker network yang tersedia

COMMAND: docker network ls



```
root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~
inet 192.168.2.76 netmask 255.255.255.0 broadcast 192.168.2.255
inet6 fe80::3c42:6d72:cf30:dae9 prefixlen 64 scopeid 0x20<link>
ether c6:e5:16:e7:7a:83 txqueuelen 1000 (Ethernet)
RX packets 409452 bytes 400354284 (400.3 MB)
RX errors 0 dropped 4707 overruns 0 frame 0
TX packets 89711 bytes 7709475 (7.7 MB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 3349 bytes 353901 (353.9 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 3349 bytes 353901 (353.9 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# docker network ls
NETWORK ID     NAME      DRIVER      SCOPE
ff4f28732eb5   bridge    bridge      local
e7775baa391b   composetest_default bridge      local
5e841a785926   host      host       local
d5913ae7df5    none      null       local
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~#
```

Figure 8.1: Cek docker network

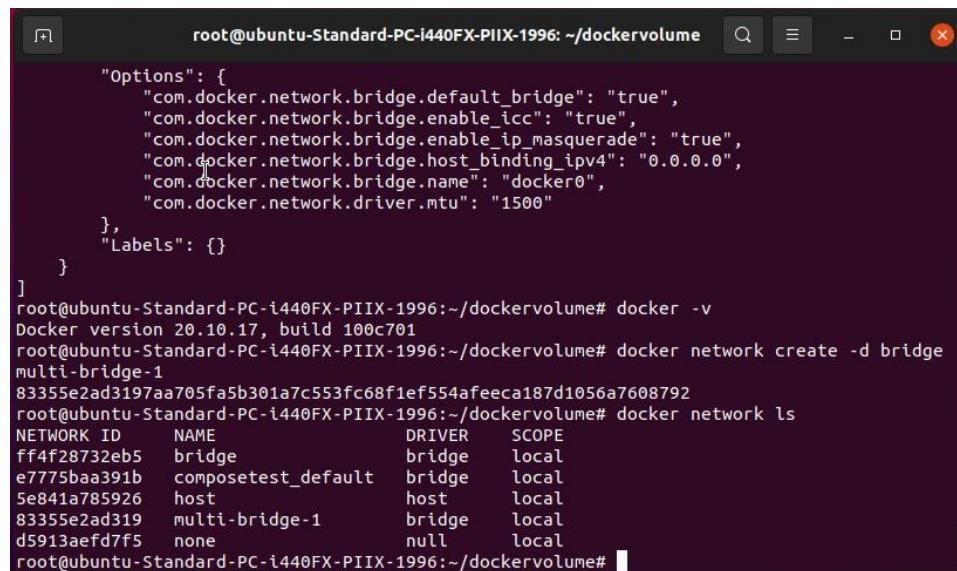
2. Buat docker network yang baru menggunakan driver bridge

Buat docker network : docker network create <DRIVER> <NAMA NETWORK>

COMMAND: docker network create -d bridge multi-bridge-1

lalu cek docker network :

COMMAND: docker network ls



```
root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~/dockervolume
{
    "Options": {
        "com.docker.network.bridge.default_bridge": "true",
        "com.docker.network.bridge.enable_icc": "true",
        "com.docker.network.bridge.enable_ip_masquerade": "true",
        "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
        "com.docker.network.bridge.name": "docker0",
        "com.docker.network.driver.mtu": "1500"
    },
    "Labels": {}
}
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockervolume# docker -v
Docker version 20.10.17, build 100c701
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockervolume# docker network create -d bridge
multi-bridge-1
83355e2ad3197aa705fa5b301a7c553fc68f1ef554afeeca187d1056a7608792
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockervolume# docker network ls
NETWORK ID     NAME      DRIVER      SCOPE
ff4f28732eb5   bridge    bridge      local
e7775baa391b   composetest_default bridge      local
5e841a785926   host      host       local
83355e2ad319   multi-bridge-1   bridge      local
d5913ae7df5    none      null       local
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~/dockervolume#
```

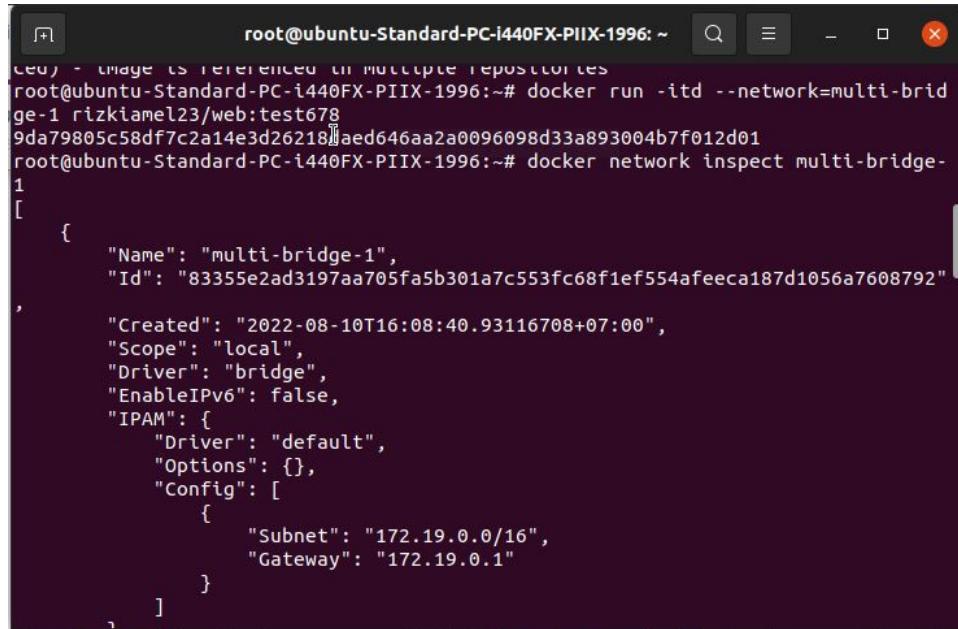
Figure 8.2: Buat docker network

## 8.2. LANGKAH MENGHUBUNGKAN LEBIH DARI 1 APLIKASI DENGAN DOCKER NETWORK41

- Buat container baru dan masukan dalam docker network yang sudah dibuat

Buat container baru dan masukan dalam docker network : docker run -itd --network=<NAMA DOCKER NETWORK> <NAMA IMAGE>

COMMAND: docker run -itd --network=multi-bridge-1 rizki-amel23/web:test678



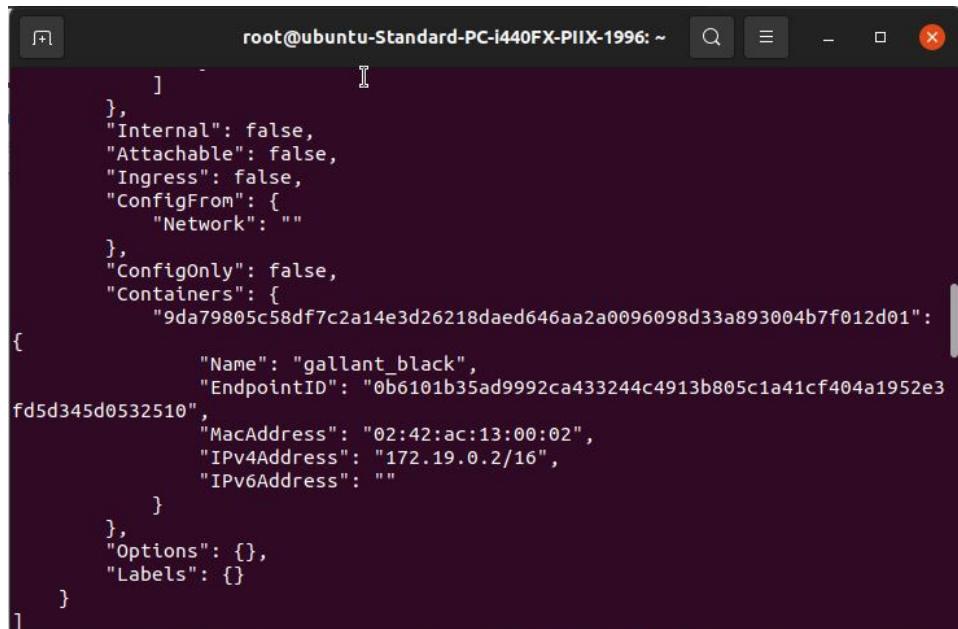
```
root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# docker run -itd --network=multi-bridge-1 rizkiamel23/web:test678
9da79805c58df7c2a14e3d26218daed646aa2a0096098d33a893004b7f012d01
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# docker network inspect multi-bridge-1
[{"Name": "multi-bridge-1", "Id": "83355e2ad3197aa705fa5b301a7c553fc68f1ef554afeeca187d1056a7608792", "Created": "2022-08-10T16:08:40.93116708+07:00", "Scope": "local", "Driver": "bridge", "EnableIPv6": false, "IPAM": {"Driver": "default", "Options": {}, "Config": [{"Subnet": "172.19.0.0/16", "Gateway": "172.19.0.1"}]}}, {"Name": "gallant_black", "EndpointID": "0b6101b35ad9992ca433244c4913b805c1a41cf404a1952e3fd5d345d0532510", "MacAddress": "02:42:ac:13:00:02", "IPv4Address": "172.19.0.2/16", "IPv6Address": ""}]}
```

Figure 8.3: Masukan container ke docker network

- Lakukan inspect pada docker network yang sudah dibuat

Lakukan inspect network : docker network inspect <NAMA DOCKER NETWORK>

COMMAND: docker network inspect multi-bridge-1



```
[{"Name": "multi-bridge-1", "Id": "83355e2ad3197aa705fa5b301a7c553fc68f1ef554afeeca187d1056a7608792", "Created": "2022-08-10T16:08:40.93116708+07:00", "Scope": "local", "Driver": "bridge", "EnableIPv6": false, "IPAM": {"Driver": "default", "Options": {}, "Config": [{"Subnet": "172.19.0.0/16", "Gateway": "172.19.0.1"}]}}, {"Name": "gallant_black", "EndpointID": "0b6101b35ad9992ca433244c4913b805c1a41cf404a1952e3fd5d345d0532510", "MacAddress": "02:42:ac:13:00:02", "IPv4Address": "172.19.0.2/16", "IPv6Address": ""}]}
```

Figure 8.4: Inspect docker network

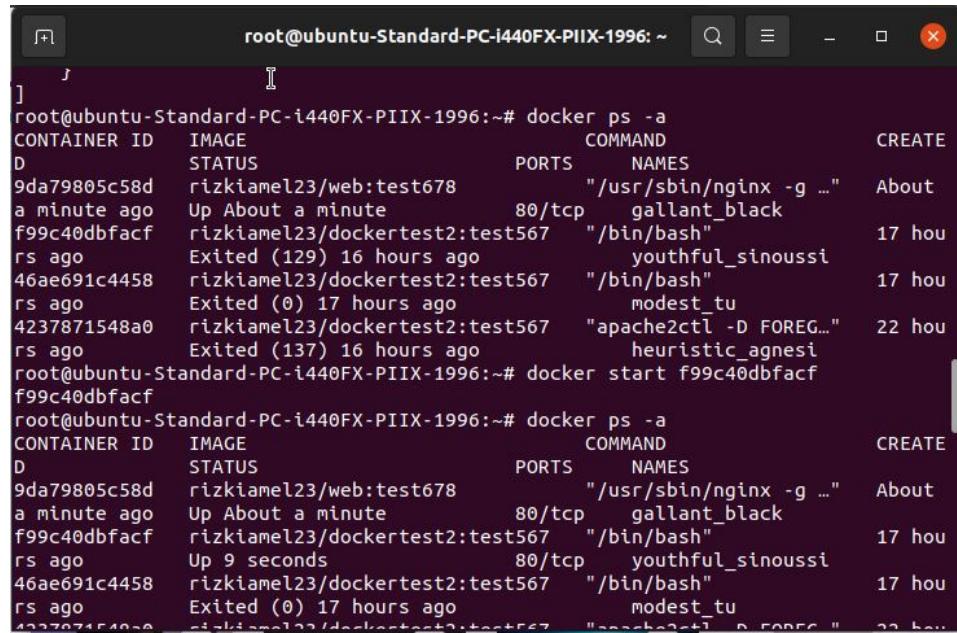
## 42 8. MENGHUBUNGKAN LEBIH DARI 1 APLIKASI DENGAN DOCKER NETWORK

Dari gambar diatas dapat dilihat container yang dibuat tadi masuk ke dalam network docker multi-bridge-1

5. Jalankan container lainnya

Jalankan container : docker start <ID CONTAINER>

COMMAND: docker start f99c40dbfaed



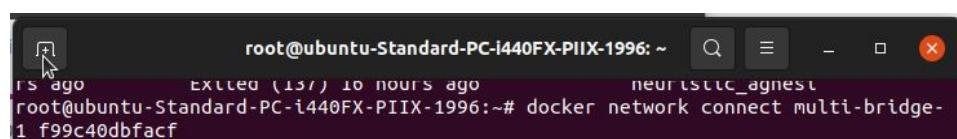
```
[root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
9da79805c58d        rizkiamel23/web:test678   "/usr/sbin/nginx -g ..."   About a minute ago   Up About a minute   80/tcp                gallant_black
f99c40dbfacf        rizkiamel23/dockertest2:test567  "/bin/bash"            17 hours ago       Exited (129) 16 hours ago   youthful_sinoussi
46ae691c4458        rizkiamel23/dockertest2:test567  "/bin/bash"            17 hours ago       Exited (0) 17 hours ago    modest_tu
4237871548a0        rizkiamel23/dockertest2:test567  "apache2ctl -D FORE..."  22 hours ago       Exited (137) 16 hours ago   heuristic_agnesi
root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~# docker start f99c40dbfaed
f99c40dbfacf
root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
9da79805c58d        rizkiamel23/web:test678   "/usr/sbin/nginx -g ..."   About a minute ago   Up About a minute   80/tcp                gallant_black
f99c40dbfacf        rizkiamel23/dockertest2:test567  "/bin/bash"            17 hours ago       Up 9 seconds       80/tcp                youthful_sinoussi
46ae691c4458        rizkiamel23/dockertest2:test567  "/bin/bash"            17 hours ago       Exited (0) 17 hours ago    modest_tu
4237871548a0        rizkiamel23/dockertest2:test567  "apache2ctl -D FORE..."  22 hours ago       Exited (137) 16 hours ago   heuristic_agnesi
```

Figure 8.5: Jalankan kontainer

6. Tambahkan container ke docker network saat container sedang berjalan

Tambahkan container ke docker network saat berjalan : docker network connect <NAMA DOCKER NETWORK> <ID CONTAINER>

COMMAND: docker network connect multi-bridge-1 f99c40dbfaed



```
[root@ubuntu-Standard-PC-i440FX-PIIX-1996: ~]# docker network connect multi-bridge-1 f99c40dbfaef
```

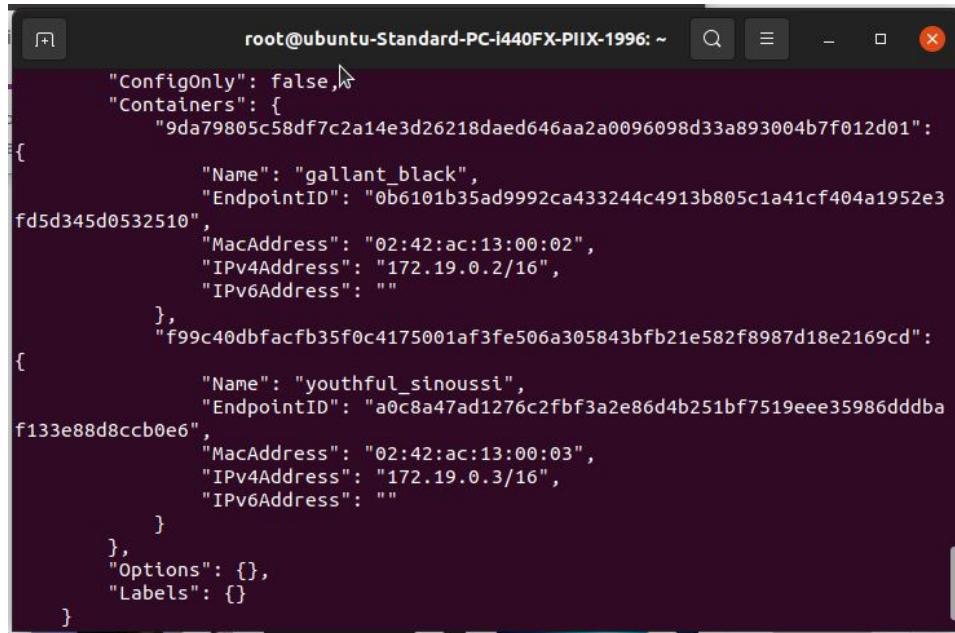
Figure 8.6: Tambah container ke docker network saat berjalan

## 8.2. LANGKAH MENGHUBUNGKAN LEBIH DARI 1 APLIKASI DENGAN DOCKER NETWORK43

### 7. Lakukan inspect pada docker network

Lakukan inspect network : docker network inspect <NAMA DOCKER NETWORK>

COMMAND: docker network inspect multi-bridge-1



```
"ConfigOnly": false,
"Containers": {
    "9da79805c58df7c2a14e3d26218daed646aa2a0096098d33a893004b7f012d01": {
        "Name": "gallant_black",
        "EndpointID": "0b6101b35ad9992ca433244c4913b805c1a41cf404a1952e3fd5d345d0532510",
        "MacAddress": "02:42:ac:13:00:02",
        "IPv4Address": "172.19.0.2/16",
        "IPv6Address": ""
    },
    "f99c40dbfacfb35f0c4175001af3fe506a305843bf21e582f8987d18e2169cd": {
        "Name": "youthful_sinoussi",
        "EndpointID": "a0c8a47ad1276c2fbf3a2e86d4b251bf7519eee35986ddba f133e88d8ccb0e6",
        "MacAddress": "02:42:ac:13:00:03",
        "IPv4Address": "172.19.0.3/16",
        "IPv6Address": ""
    }
},
"Options": {},
"Labels": {}
}
```

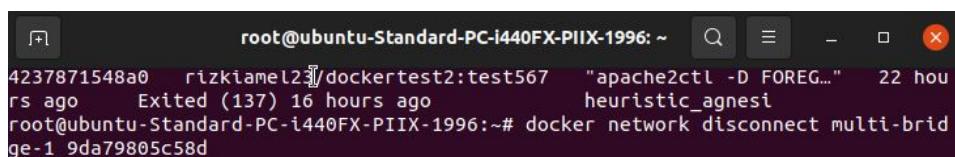
Figure 8.7: Inspect docker network

Dari gambar diatas dapat dilihat container yang dijalankan tadi telah ditambahkan pada docker network multi-bridge-1, sehingga kedua aplikasi dapat berkomunikasi dalam jaringan tersebut

### 8. Bila ingin memutuskan container dari network

Memutuskan container dari network : docker network disconnect <NAMA DOCKER NEWORK> <ID CONTAINER>

COMMAND: docker network disconnect multi-bridge-1 9da79805c58d

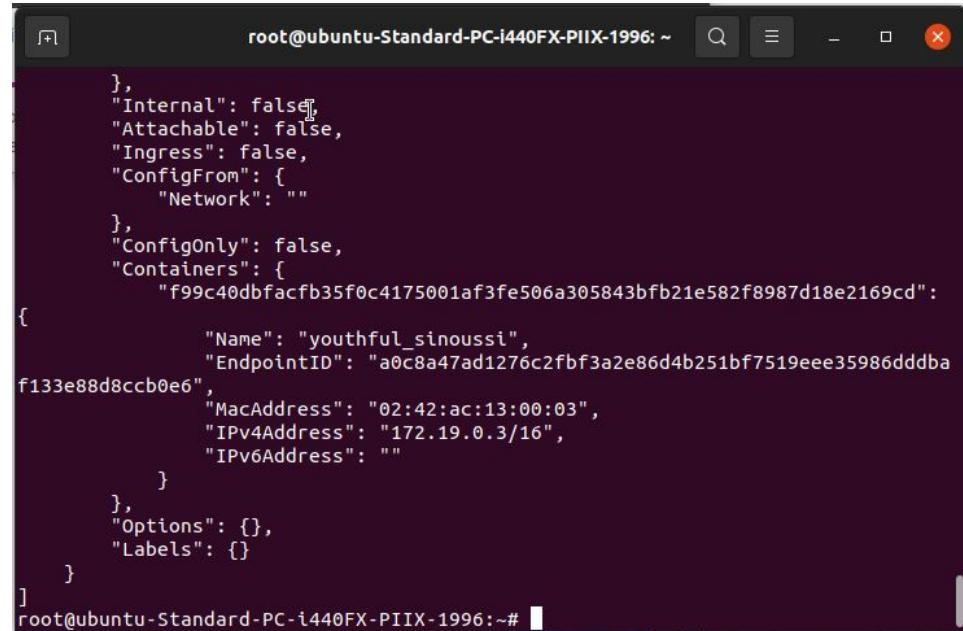


```
4237871548a0 rizkiamel2@dockertest2: test567 "apache2ctl -D FORE..." 22 hours ago Exited (137) 16 hours ago heuristic_agensi
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# docker network disconnect multi-bridge-1 9da79805c58d
```

Figure 8.8: Memutuskan docker network

#### 44 8. MENGHUBUNGKAN LEBIH DARI 1 APLIKASI DENGAN DOCKER NETWORK

Sehingga saat dilakukan inspect multi-bridge-1 list container berkurang



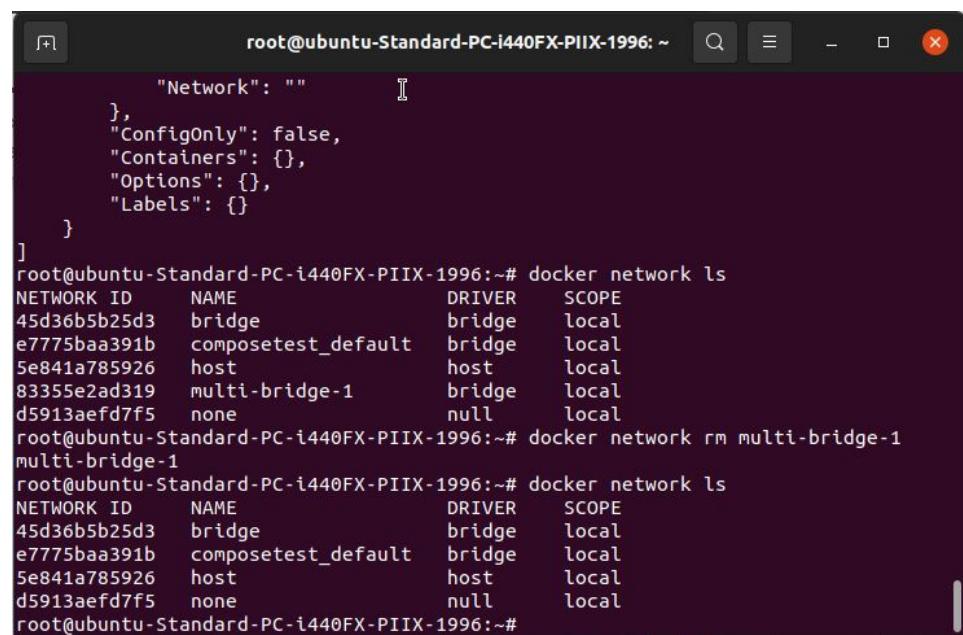
```
[{"Name": "multi-bridge-1", "Id": "d5913aef7f5", "Created": "2024-01-11T10:45:12.000000Z", "Scope": "local", "Driver": "bridge", "Config": {"Internal": false, "Attachable": false, "Ingress": false, "ConfigFrom": {}, "Network": ""}, "Containers": [{"Name": "youthful_sinoussi", "EndpointID": "a0c8a47ad1276c2fbf3a2e86d4b251bf7519eee35986ddba", "MacAddress": "02:42:ac:13:00:03", "IPv4Address": "172.19.0.3/16", "IPv6Address": ""}], "Options": {}, "Labels": {}}]
```

Figure 8.9: Inspect docker network

9. Bila ingin menghapus docker network

Menghapus docker network : docker network rm <NAMA DOCKER NEWORK>

COMMAND: docker network rm multi-bridge-1



```
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
45d36b5b25d3    bridge    bridge      local
e7775baa391b    compositetest_default  bridge      local
5e841a785926    host      host      local
83355e2ad319   multi-bridge-1    bridge      local
d5913aef7f5     none      null      local
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# docker network rm multi-bridge-1
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
45d36b5b25d3    bridge    bridge      local
e7775baa391b    compositetest_default  bridge      local
5e841a785926    host      host      local
d5913aef7f5     none      null      local
```

Figure 8.10: Menghapus docker network

# 9

## Perbedaan Docker Version

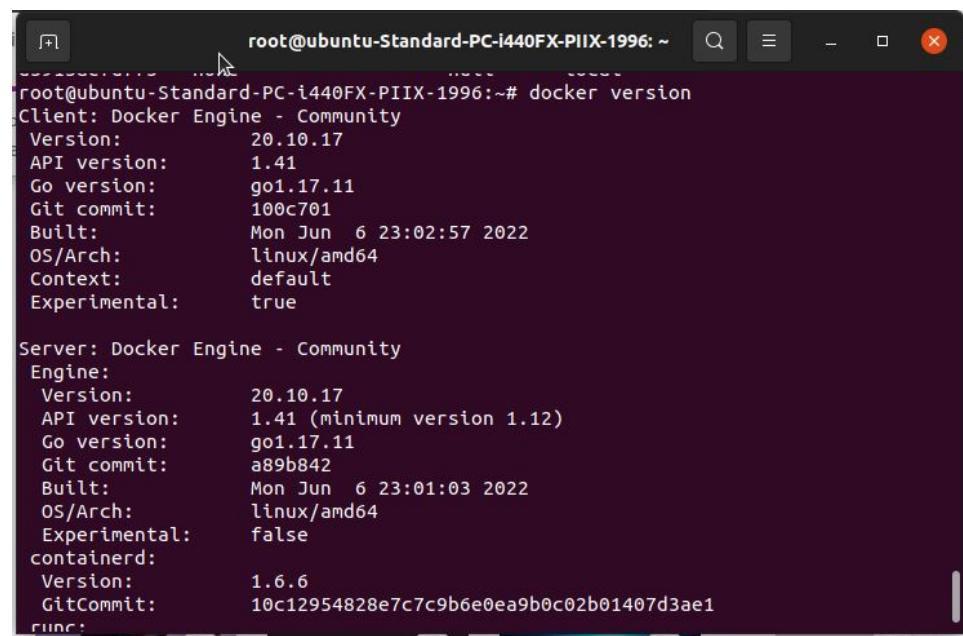
### 9.1 Pembuka

Docker mulai dirilis pada tahun 2013 dimulai dari versi 0 hingga saat ini docker telah mencapai versi 20. Pada setiap update versi nya docker memberikan fitur baru atau perbaikan dari versi rilisan sebelumnya. Detail rilisan dapat dilihat di website dokumentasi docker

<https://docs.docker.com/engine/release-notes/>.

Jika ingin mengecek versi docker melalui command line :

COMMAND: docker version



```
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# docker version
Client: Docker Engine - Community
  Version: 20.10.17
  API version: 1.41
  Go version: go1.17.11
  Git commit: 100c701
  Built: Mon Jun 6 23:02:57 2022
  OS/Arch: linux/amd64
  Context: default
  Experimental: true

Server: Docker Engine - Community
  Engine:
    Version: 20.10.17
    API version: 1.41 (minimum version 1.12)
    Go version: go1.17.11
    Git commit: a89b842
    Built: Mon Jun 6 23:01:03 2022
    OS/Arch: linux/amd64
    Experimental: false
  containerd:
    Version: 1.6.6
    GitCommit: 10c12954828e7c7c9b6e0ea9b0c02b01407d3ae1
  runc:
```

Figure 9.1: Cek versi docker

## 9.2 Perbedaan docker version

Berikut perbedaan yang dari versi docker 1 - versi 20 :

#	Versi 1	Versi 17	Versi 18	Versi 19	Versi 20
Tahun Rilis	2014-2017	2018	2018-2019	2021	2020-2022
Builder	Support for comperessing build, step number, network	Support for ADD urls without any sub path	Added prune options to the API	Beta versions of apparmor are now parsed correctly preventing build failures	Updated the bundled version of buildx to v0.8.0
Networking	Support attachable network, windows server 2016 overlay, sepecifying IP Address	Support verbose info to partial overlay ID	Added SSH agent socket forwarder when using BuildKit	Disable IPv6 Router Advertisements to prevent address spoofing.	Update libnetwork to fix publishing ports on environments with kernel boot
API	Support secrets in docker stack deploy with compose file	Support for proxy configuration in config.json	Updated API version to 1.39	Updated API version to v1.40.	Update API version to v1.41
Plugins	Support global scoped network plugins, upgrade plugin	Make plugin removes more resilient to failure	Configured container log-level to be the same as dockerd	Added basic framework for writing and running CLI plugins	Socker plugin create making compatible with older versions of Docker
Security	Selinux labeling of volumes shared in a container	Redact secret data on secret creation	Added support for build-time secrets when using BuildKit	Prevent an invalid image from crashing docker daemon	Profiles: seccomp: update to Linux 5.11 syscall list



# 10

## Tentang Docker Hub

### 10.1 Pengertian

Dikutip dari situs Docker Docs, Docker Hub merupakan layanan yang disediakan oleh docker untuk menemukan dan berbagi docker images. Docker Hub merupakan gudang docker images di dunia dari berbagai sumber baik dari developer, open source project, dan software vendor. Pengguna dapat mengakses repositori di docker hub secara publik untuk menyimpan dan berbagi docker image.

### 10.2 Fitur Docker Hub

- Repositories : menyimpan dan berbagi docker images.
- Team dan Organizations : mengatur akses publik/privatnya repositori untuk diakses oleh tim.
- Docker Official Images : Berbagi docker images dari sumber docker.
- Docker Verified Publisher Images : Berbagi docker images dari sumber vendor eksternal.
- Builds : Buat image container secara otomatis dari Github dan Bitbucket dan simpan di docker hub.
- Webhooks : Memicu tindakan setelah push berhasil ke repositori untuk mengintegrasikan Docker Hub dengan layanan lain.
- Docker menyediakan Docker Hub CLI tool dan API untuk mengizinkan pengguna berinteraksi dengan docker hub.