

PETEMUAN I

SETUP LINUX SERVER DAN AUTO DEPLOY

DENGAN DOCKER DAN GITHUB

Linux

- **Linux**

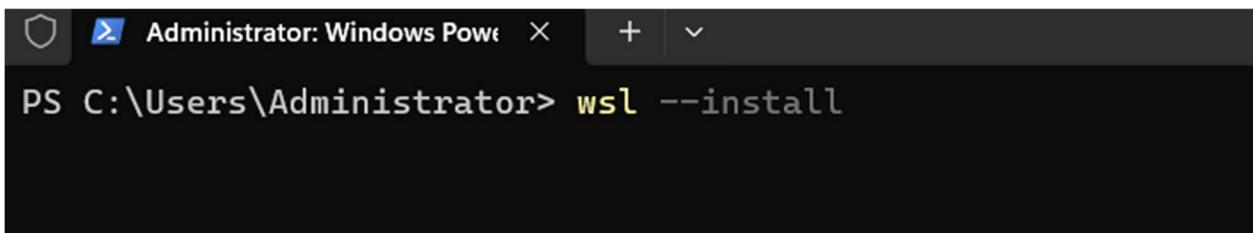
Linux adalah sistem operasi open-source yang sangat populer digunakan untuk server, desktop, dan berbagai perangkat lain. Linux berbasis command line interface (CLI) yang memungkinkan kita mengontrol sistem melalui perintah-perintah di terminal.

- **WSL**

WSL atau Windows Subsystem for Linux adalah sebuah fitur di Windows yang memungkinkan kita menjalankan sistem operasi Linux secara langsung tanpa perlu menggunakan virtual machine. Dengan WSL, kita bisa membuka terminal Linux dan menjalankan perintah Linux layaknya sedang berada di server Linux asli. Dalam materi ini, kita akan menggunakan WSL sebagai “server lokal” kita. Dengan WSL, kita bisa belajar bagaimana cara menggunakan Linux, menjalankan Docker, dan melakukan deployment aplikasi secara langsung di lingkungan yang mirip dengan server sesungguhnya.

Instalasi WSL :

- Open Terminal/CMD
- Ketikan ”wsl --install” dan tunggu proses sampai selesai



```
PS C:\Users\Administrator> wsl --install
```

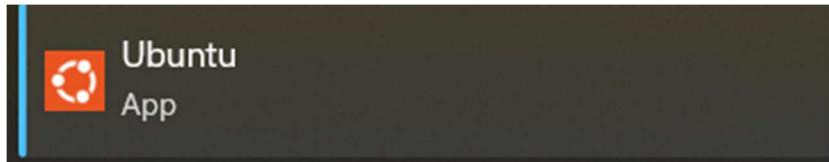
Ikuti langkah step yang berjalan sampai selesai.

```
NAME          STATE      VERSION
* docker-desktop Stopped      2
PS C:\Users\Administrator> wsl --install
Downloading: Ubuntu
Installing: Ubuntu
Distribution successfully installed. It can be launched via 'wsl.exe -d Ubuntu'
Launching Ubuntu...
Provisioning the new WSL instance Ubuntu
This might take a while...
Create a default Unix user account: bagja
New password:
Retype new password:
```

Setelah selesai instalasi WSL nya akan langsung masuk ke WSL sistem linux Ubuntu

```
bagja@WIN-STQV5VA578G:\mnt\c\Users\Administrator$ ls
'Application Data'  'My Documents'
'Contacts'          'NTUSER.DAT'
'Cookies'           'NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TM.blf'
'Desktop'            'NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TMContainer000000000000000000000001.regtrans-ms'
'Documents'          'NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TMContainer000000000000000000000002.regtrans-ms'
'Downloads'          'NetHood'
'Favorites'          'OneDrive'
'Links'              'Pictures'
'Local Settings'    'Postman'
bagja@WIN-STQV5VA578G:\mnt\c\Users\Administrator$ cd
bagja@WIN-STQV5VA578G:~$ ks
ks: command not found
bagja@WIN-STQV5VA578G:~$ ls
bagja@WIN-STQV5VA578G:~$
```

- Untuk membuka WSL di awal, kamu bisa mencari aplikasi dengan mengetikkan "Ubuntu" pada fitur pencarian (search) di Windows.



- **Struktur File System Linux**

Linux menggunakan sistem file yang berbentuk hierarki, dimulai dari direktori paling atas yang disebut root directory, ditandai dengan tanda garis miring /. Semua file dan folder lain berada di bawah root ini dalam susunan yang terorganisir seperti pohon.

Berikut beberapa direktori penting yang harus diketahui:

- **/(Root Directory)**
Ini adalah direktori paling atas dalam sistem Linux. Semua file dan folder, termasuk folder sistem dan data pengguna, berawal dari sini.
 - **/home**
Tempat penyimpanan data dan file milik pengguna (user). Setiap pengguna di sistem biasanya memiliki folder sendiri di dalam /home, misalnya /home/username.
 - **/etc**
Direktori yang berisi file konfigurasi sistem dan aplikasi. Contohnya konfigurasi jaringan, konfigurasi layanan, dan setelan lainnya yang dibutuhkan sistem.
 - **/var**
Direktori ini menyimpan file yang bersifat variabel, seperti log sistem, database sementara, dan file aplikasi yang berubah-ubah selama sistem berjalan.
 - **/usr**
Tempat penyimpanan aplikasi, library, dan file data yang dibutuhkan program-program pengguna. Direktori ini berisi program-program yang sudah terinstall dan file-file yang tidak berubah secara rutin.

- /bin dan /sbin

Berisi program-program eksekusi penting yang dibutuhkan sistem untuk berjalan.

- o /bin biasanya berisi perintah-perintah dasar untuk semua pengguna, seperti ls, cp, mv.
- o /sbin berisi program yang digunakan oleh administrator sistem (root) untuk mengelola sistem.

- **Navigasi Dasar Dalam Terminal**

Saat bekerja di Linux, kita berinteraksi dengan sistem melalui **terminal** menggunakan perintah-perintah untuk mengakses dan mengelola file serta folder. Berikut beberapa perintah navigasi dasar yang wajib di kuasai:

Perintah	Fungsi	Contoh Penggunaan
pwd	Menampilkan lokasi direktori saat ini	Ketik pwd dan tekan Enter untuk melihat di folder mana kamu sedang berada sekarang.
ls	Menampilkan daftar file dan folder	Ketik ls untuk melihat isi folder. Tambahkan opsi -l (ls -l) untuk melihat informasi detail seperti ukuran, tanggal, dan permission.
cd	Pindah ke direktori tertentu	Gunakan cd diikuti path folder tujuan, misalnya cd /home/user untuk masuk ke folder user.
cd ~	Kembali ke direktori home user	Perintah cd ~ akan membawa kamu ke direktori home (folder pribadi) user saat ini, misalnya /home/username.
cd ..	Pindah ke direktori induk (folder satu tingkat di atas)	Gunakan cd .. untuk naik satu tingkat ke folder induk.

- **Manipulasi File dan Folder**

Setelah kita mengetahui cara berpindah-pindah direktori, langkah selanjutnya adalah memahami bagaimana cara membuat, mengedit, menyalin, memindahkan, dan menghapus file serta folder di Linux. Semua kegiatan ini dilakukan melalui terminal dengan menggunakan perintah-perintah sederhana.

Berikut ini beberapa perintah dasar yang sangat penting:

Perintah	Fungsi	Contoh Penggunaan
mkdir nama_folder	Membuat folder baru	mkdir project
touch nama_file	Membuat file kosong	touch file.txt
nano nama_file	Mengedit file menggunakan editor teks sederhana di terminal	nano index.html
cat nama_file	Menampilkan isi file ke terminal	cat README.md
cp sumber tujuan	Menyalin file atau folder ke lokasi lain	cp file.txt backup.txt
mv sumber tujuan	Memindahkan atau mengganti nama file/folder	mv file.txt ../file_lama.txt
rm nama_file	Menghapus file	rm file.txt
rm -r nama_folder	Menghapus folder beserta seluruh isinya	rm -r old_folder

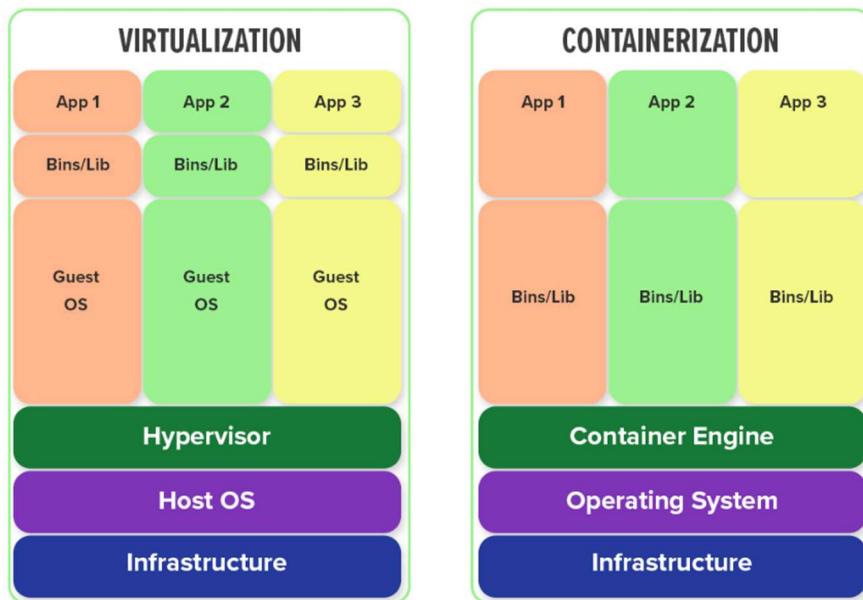
Docker

Virtualization vs Containerization

Pengertian Virtualization dan Containerization

Virtualization adalah teknologi yang memungkinkan pembuatan mesin virtual di dalam satu fisik server. Dengan menggunakan hypervisor, virtualisasi memungkinkan pengelolaan beberapa sistem operasi atau aplikasi yang berjalan secara mandiri. Konsep dasar virtualisasi melibatkan isolasi sumber daya antara mesin virtual, sehingga setiap mesin virtual dapat beroperasi seolah-olah menjadi mesin fisik yang terpisah.

Sedangkan Containerization adalah teknologi yang memungkinkan pengemasan aplikasi dan dependensinya ke dalam sebuah wadah (container) yang dapat dijalankan secara konsisten di berbagai lingkungan komputasi, tanpa perlu mengubah kode atau konfigurasi aplikasi itu sendiri. Container merupakan unit yang portabel, ringan, dan dapat diisolasi, yang mengemas aplikasi, library, dan konfigurasi menjadi satu entitas yang dapat dijalankan di lingkungan yang berbeda, seperti lokal, cloud, atau pusat data.



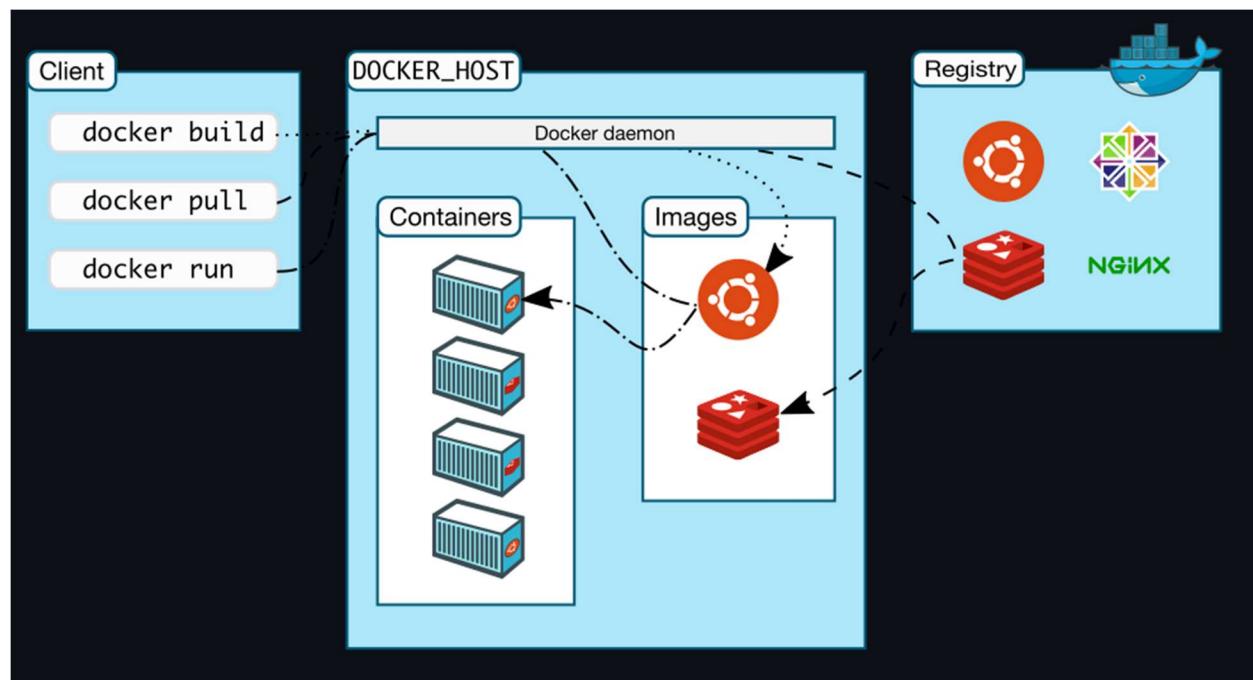
- Virtualisasi menggunakan hypervisor untuk membuat mesin virtual yang memerlukan sistem operasi penuh dan isolasi sumber daya seperti CPU, RAM, dan storage untuk setiap mesin virtual. Sementara itu, containerization menggunakan teknologi seperti Docker untuk membuat wadah (container) yang berbagi sistem operasi host.

- Virtualisasi memungkinkan menjalankan sistem operasi dan aplikasi yang berbeda secara simultan dalam mesin virtual yang terisolasi. Sementara itu, containerization memungkinkan menjalankan aplikasi yang dikemas dalam container di dalam host yang sama, berbagi kernel OS yang sama.
- Virtualisasi cenderung lebih cocok untuk aplikasi yang membutuhkan isolasi penuh, konfigurasi yang kompleks, dan dukungan untuk berbagai sistem operasi. Di sisi lain, containerization lebih cocok untuk aplikasi yang bersifat ringan, portabel, dan bisa dijalankan di berbagai lingkungan komputasi.
- Proses start-up pada virtualisasi memerlukan waktu yang lebih lama, karena melibatkan booting sistem operasi dan konfigurasi tambahan pada setiap mesin virtual. Containerization, di sisi lain, memungkinkan proses deploy dan start-up yang lebih cepat, karena hanya perlu menjalankan container yang sudah dikemas dan siap dijalankan.

Pengertian Docker

Docker adalah sebuah platform yang memungkinkan pengembang perangkat lunak untuk membuat, mengemas, dan menjalankan aplikasi dalam wadah yang dapat diisolasi secara mandiri, disebut container. Container dalam Docker berfungsi seperti lingkungan eksekusi yang terisolasi untuk menjalankan aplikasi, termasuk kode sumber, runtime, dan dependensi yang diperlukan.

Arsitektur Docker



Docker Daemon

Docker Daemon adalah komponen yang berjalan di latar belakang (background) pada host dan bertanggung jawab untuk menjalankan dan mengelola Docker Object seperti images, container, network, dan lain-lain. Docker Daemon adalah proses yang berjalan di dalam sistem operasi host dan menerima perintah dari Docker Client untuk membuat, menjalankan, menghentikan, dan mengelola Docker Object. Docker Daemon juga bertanggung jawab untuk mengelola sumber daya host seperti CPU, memori, dan jaringan yang digunakan oleh Docker Object.

Docker Client

Docker Client adalah antarmuka pengguna berbasis command-line atau GUI yang digunakan untuk berinteraksi dengan Docker. Docker Client memungkinkan pengguna untuk menjalankan perintah-perintah Docker untuk membuat, mengelola, dan mengontrol layanan pada Docker. Docker Client berkomunikasi dengan Docker Daemon untuk mengirimkan perintah-perintah Docker dan menerima output layanan Docker yang sedang berjalan.

Docker Objects

Docker Objects adalah komponen dasar yang terdapat di Docker. Beberapa contoh Docker Objects meliputi image, container, volume, dan network.

Docker Registry

Docker Registry adalah repositori yang digunakan untuk menyimpan dan berbagi Docker Image. Docker Registry berfungsi sebagai tempat penyimpanan untuk Docker Image yang dapat diakses oleh pengguna Docker dari berbagai lokasi. Docker Hub, yang merupakan Docker public registry, adalah salah satu contoh Docker Registry yang sering digunakan untuk menyimpan dan berbagi Docker Image secara publik. Selain Docker Hub, pengguna juga dapat membuat Docker Registry pribadi untuk menyimpan Docker Image.

Instalasi Docker dan Docker Compose

```
# Install dependensi awal
sudo apt-get update
sudo apt-get install -y ca-certificates curl
# Tambahkan GPG key Docker
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
# Tambahkan repository Docker
```

```
echo \  
"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg]  
https://download.docker.com/linux/ubuntu \  
"$(./etc/os-release && echo "$VERSION_CODENAME")" stable" | \  
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  
# Update package list  
sudo apt-get update  
# Install paket yang diperlukan  
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-  
compose-plugin  
# Verifikasi instalasi  
docker compose version  
# 1. Tambahkan user ke grup docker  
sudo usermod -aG docker $USER  
# 2. Aktifkan perubahan grup (tanpa logout)  
newgrp docker  
# 3. Verifikasi  
docker ps # Sekarang harus bekerja tanpa sudo
```

Instalasi Golang

```
sudo apt update  
sudo wget https://go.dev/dl/go1.24.3.linux-amd64.tar.gz  
sudo tar -C /usr/local -xzf go1.24.3.linux-amd64.tar.gz  
export PATH=$PATH:/usr/local/go/bin  
source ~/.profile  
#cek versio golang  
go version  
#Daftarkan Golang ke sudo  
sudo visudo  
(Cari yang model perintahnya seperti ini pada dalam config cisudo tersebut “Defaults  
secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin” ” )
```

```
Defaults      env_reset
Defaults      mail_badpass
Defaults      secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
Defaults      use_pty
```

Tempelkan config ini di bawahnya

```
Defaults
secure_path="/usr/local/go/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

Defaults      env_reset
Defaults      mail_badpass
Defaults      secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/snap/bin"
Defaults      use_pty
Defaults      secure_path="/usr/local/go/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
```

#cek dengan sudo

Sudo go version

Docker Image

Docker Image bisa diibaratkan seperti installer sebuah aplikasi. Di dalamnya sudah terdapat aplikasi utama beserta seluruh dependency (perpustakaan, konfigurasi, dan sistem pendukung) yang dibutuhkan agar aplikasi tersebut bisa berjalan dengan baik di lingkungan Docker.

Berikut comand dasar nya

<code>docker images</code>	Melihat daftar image
<code>docker pull <image></code>	Menarik image dari Docker Hub
<code>docker rmi <image></code>	Menghapus image
<code>docker tag <image> <new></code>	Memberi nama/alias pada image
<code>docker build -t <name> .</code>	Build image dari Dockerfile

Docker Container

Jika Docker Image diibaratkan seperti installer aplikasi, maka Docker Container adalah aplikasi yang sudah terinstal dan berjalan dari installer tersebut.

Satu Docker Image bisa digunakan untuk membuat beberapa Docker Container, selama masing-masing container memiliki nama yang berbeda.

Perlu diketahui, Docker Image yang sedang digunakan oleh container tidak bisa dihapus, karena Docker Container tidak menyalin isi image tersebut, melainkan menggunakan langsung isi dari image saat dijalankan.

docker run <image>	Menjalankan container baru dari image
docker run -it <image>	Interaktif (biasanya untuk shell)
docker run -d <image>	Menjalankan container di background
docker run --name <name> <image>	Menamai container
docker run -p 8080:80 <image>	Port mapping host:container
docker run -v /host:/cont <image>	Mount volume
docker run --env VAR=value <image>	Environment variable
docker ps	Melihat container yang sedang berjalan
docker ps -a	Semua container (termasuk yang berhenti)
docker start <name>	Menjalankan ulang container
docker stop <name>	Menghentikan container
docker restart <name>	Restart container
docker rm <name>	Menghapus container
docker exec -it <name> bash	Masuk ke terminal container
docker logs <name>	Melihat log container
docker inspect <name>	Detail konfigurasi container

Docker Network

Secara default, saat kita membuat container di Docker, setiap container akan terisolasi satu sama lain. Artinya, container tidak bisa saling berkomunikasi secara langsung tanpa konfigurasi tambahan.

Untuk mengatasi hal ini, Docker menyediakan fitur yang disebut Docker Network.

Dengan menggunakan Docker Network, kita dapat membuat jaringan virtual di dalam Docker, dan menghubungkan beberapa container ke dalam jaringan yang sama.

Jika beberapa container berada di dalam network yang sama, maka mereka dapat berkomunikasi satu sama lain secara otomatis, cukup dengan menyebut nama container sebagai alamatnya.

```
docker network ls           Melihat daftar jaringan  
docker network create <name>  Membuat network  
docker network inspect <name> Detail konfigurasi network  
docker network rm <name>    Menghapus network  
docker run --network <name>  Gunakan network tertentu
```

Mencoba untuk menarik image dari docker hub

```
#contoh untuk menarik image mysql  
docker pull mysql  
  
#contoh untuk menarik image phpMyAdmin  
docker pull phpmyadmin/phpMyAdmin
```

Melihat list images yang sudah di Tarik/download

```
docker images
```

```
kelas-santai@kelas-santai:~$ docker images  
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE  
redis               latest   860da63e75fb  10 days ago  128MB  
mysql               latest   edbdd97bf78b  7 weeks ago  859MB  
phpmyadmin/phpmyadmin  latest   0276a66ce322  4 months ago  571MB  
kelas-santai@kelas-santai:~$ |
```

Menjalankan Image yang sudah di download atau tarik

Menjalankan image mysql

```
docker run -d \  
--name mysql-container \  
-e MYSQL_ROOT_PASSWORD=rootpassword \  
-e MYSQL_DATABASE=contohdb \  
-e MYSQL_USER=contohuser \  
-e MYSQL_PASSWORD=contohpass \  
-p 3306:3306 \  
mysql:latest
```

Penjelasan

docker run -d \

- **docker run**: Perintah untuk menjalankan container baru.
- **-d (detached mode)**: Container akan berjalan di background (tidak mengunci terminal).

--name mysql-container \

- Memberi nama container: mysql-container (agar mudah dikelola, seperti docker stop mysql-container).

-e MYSQL_ROOT_PASSWORD=rootpassword \

- Set password user **root** MySQL menjadi rootpassword.

-e MYSQL_DATABASE=contohdb \

- Secara otomatis membuat database dengan nama contohdb saat container pertama kali dibuat.

-e MYSQL_USER=contohuser \

- Membuat user baru MySQL dengan nama contohuser.

-e MYSQL_PASSWORD=contohpass \

- Set password untuk user contohuser menjadi contohpass.

-p 3306:3306 \

- **Port mapping** dari:

- Port 3306 di container (default port MySQL)
- Ke port 3306 di host (agar bisa diakses dari luar)

mysql:latest

- Menentukan image yang digunakan, yaitu mysql versi terbaru
- Jika image belum ada, Docker akan otomatis **menarik image dari Docker Hub**.

```
kelas-santai@kelas-santai:~$ docker run -d \
--name mysql-container \
-e MYSQL_ROOT_PASSWORD=rootpassword \
-e MYSQL_DATABASE=contohdb \
-e MYSQL_USER=contohuser \
-e MYSQL_PASSWORD=contohpass \
-p 3306:3306 \
mysql:latest
46794a6e0940b4a938491bfe2166b2c1242512ad21ee9b90a572b0a05eb39e53
kelas-santai@kelas-santai:~$ |
```

Jika Berhasil maka akan muncul seperti di atas

Masuk ke container database mysql dan membuat user baru

- Masuk ke dalam container database
docker exec -it mysql-container sh

```
kelas-santai@kelas-santai:~$ docker exec -it mysql-container sh
sh-5.1#
```

- Masuk sebagai user root untuk mysql
mysql -u root -p

masukan password yang terlah di buat ketika menjalankan kontainer

```
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)
bash-5.1# mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 9.3.0 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> ls
--> ^C
mysql> show databases;
--> ^C
mysql>
```

sudah berhasil masuk ke mysql test untuk mengecek semua database di mysql dengan perintah
SHOW DATABASES;

```
mysql> show databases;
+-----+
| Database |
+-----+
| contohdb |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.019 sec)
```

- Membuat user baru untuk phpmyadmin
#membuat user baru
CREATE USER 'admin'@'%' IDENTIFIED BY 'admin123';
#Maka akan membuat user baru "admin" dengan password "admin123"
#Memberikan hak akses ke semua database
GRANT ALL PRIVILEGES ON ** TO 'admin'@'%' WITH GRANT OPTION;
#menyimpan perubahan
FLUSH PRIVILEGES;

Menjalankan image phpmyadmin

```
docker run -d \
--name phpmyadmin-container \
--link mysql-container:mysql \
-e PMA_HOST=mysql \
-e PMA_PORT=3306 \
-e PMA_ARBITRARY=1 \
-p 8080:80 \
phpmyadmin/phpmyadmin
```

docker run -d \

- Jalankan container baru dalam mode **detached** (background)

--name phpmyadmin-container \

- Beri nama container ini phpmyadmin-container agar mudah di-manage.

**--link mysql-container:mysql **

- Hubungkan container phpmyadmin-container dengan container MySQL bernama mysql-container.
- Container phpMyAdmin bisa mengakses MySQL lewat hostname mysql (alias).

Catatan: --link sudah deprecated, tapi masih dipakai untuk kemudahan cepat koneksi antar container.

**-e PMA_HOST=mysql **

- Set environment variable PMA_HOST untuk phpMyAdmin supaya tahu hostname MySQL yang akan diakses, yaitu mysql (nama alias dari --link).

**-e PMA_PORT=3306 **

- Tentukan port MySQL, defaultnya 3306.

**-e PMA_ARBITRARY=1 **

- Mengaktifkan fitur phpMyAdmin untuk login ke server MySQL arbitrary (tidak harus hanya host yang sudah ditentukan). Bisa login ke server MySQL lain jika diperlukan.

**-p 8080:80 **

- Mapping port:
 - Port 80 di dalam container phpMyAdmin (web server)
 - Ke port 8080 di host (komputer/laptop kamu)

Jadi phpMyAdmin dapat diakses di browser dengan alamat:
<http://localhost:8080>

phpmyadmin/phpmyadmin

- Nama image yang digunakan, dari Docker Hub resmi phpMyAdmin.

Membuat Image pada aplikasi golang

Buatlah aplikasi folder dengan struktur berikut

myapps/myapp/

```

├── main.go
├── go.mod
└── go.sum

```

#membuat folder mysqpp dan msuk ke folder myapp

sudo mkdir mypp

cd myapp

```
#membuat file golang service
sudo go mod init myapp
sudo nano main.go
#isi file dari main.go
package main

import (
    "fmt"
    "net/http"
)

func main() {
    http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        fmt.Fprintln(w, "Halo dari Docker + Golang!")
    })
    fmt.Println("Server running on port 3000")
    http.ListenAndServe(":3000", nil)
}
```

```
#membuat Dockerfile didalam folder myap
sudo nano Dockerfile
```

```
#isi dari Dockerfile
#liat versi golang yang ada di go.mod
FROM golang:1.24.3-alpine

WORKDIR /app

# Copy semua source code ke dalam container
COPY ..

# Download dependensi
RUN go mod download
```

```
# Expose port (jika kamu dengarkan di 0.0.0.0:3000)
EXPOSE 3000
```

```
# Jalankan langsung dengan go run
CMD ["go", "run", "main.go"]
```

```
#Simpan dan close
```

```
#Membuat Image untuk myapp
```

```
docker build -t myapp:latest .
```

```
kelas-santai@kelas-santai:~/myapp$ docker build -t myapp:latest .
[+] Building 23.8s (9/9) FINISHED
```

```
#Jalankan container di port 3000
```

```
docker run -d -p 3000:3000 myapp:latest
```

```
#cek container yang berjalan
```

```
docker ps
```

```
kelas-santai@kelas-santai:~/myapp$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
7e3af1f2e908        myapp:latest       "go run main.go"   About a minute ago   Up About a minute   0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp          mystifying_shtern
0d721d6f6adb        phpmyadmin/phpmyadmin   "/docker-entrypoint..."   About an hour ago    Up About an hour    0.0.0.0:8080->80/tcp, [::]:8080->80/tcp          phpmyadmin-container
46794a6e0940        mysql:latest        "/docker-entrypoint.s..."   About an hour ago    Up About an hour    0.0.0.0:3306->3306/tcp, [::]:3306->3306/tcp, 33060/tcp      mysql-container
kelas-santai@kelas-santai:~/myapp$ |
```

Cek browser :

<http://localhost:3000> atau <http://ipserver:3000>

Docker Compose

Docker Compose adalah sebuah alat atau tool untuk mengelola dan menjalankan aplikasi yang terdiri dari satu atau beberapa container. Docker Compose memungkinkan untuk mendefinisikan, mengkonfigurasi, dan menjalankan beberapa Docker Container sekaligus dengan menggunakan file konfigurasi YAML yang sederhana.

Docker Compose juga dapat menentukan Docker Image untuk setiap Docker Container, mengatur pengaturan jaringan, menentukan volume yang dibutuhkan, dan melakukan konfigurasi lainnya dalam satu file konfigurasi. Selain itu, Docker Compose juga memudahkan proses pengaturan dan penyebaran aplikasi pada lingkungan produksi atau development yang berbeda dengan cara yang konsisten.

```
docker-compose up      # Menjalankan semua service
docker-compose up -d   # Menjalankan di background (detached mode)
docker-compose down    # Menghentikan dan menghapus container
docker-compose build   # Build ulang service
docker-compose ps     # Melihat container yang sedang berjalan
docker-compose logs   # Melihat log container
```

File Konfigurasi docker-compose.yml

```
version: '3.8'
services:
  backend:
    build: ./myapp
    ports:
      - "3000:3000"
    networks:
      - elatar-network
    depends_on:
      - mysql
    restart: unless-stopped
  mysql:
    image: mysql:latest
    container_name: mysql-container
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: contohdb
      MYSQL_USER: contohuser
      MYSQL_PASSWORD: contohpass
    ports:
      - "3306:3306"
    networks:
      - elatar-network
    volumes:
      - mysql-data:/var/lib/mysql
  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    container_name: phpmyadmin-container
    restart: always
```

```
environment:
  PMA_HOST: mysql
  PMA_PORT: 3306
  PMA_ARBITRARY: 1
ports:
  - "8080:80"
depends_on:
  - mysql
networks:
  - elatar-network

networks:
  elatar-network:
    driver: bridge

volumes:
  mysql-data:

#Mencoba menjalankan container lewat docker compose
#menghentikan semua container yang erjalan
docker stop $(docker ps -aq)

#menghapus semua container
docker rm $(docker ps -aq)

#menjalankan docker compose
docker compose up -d --build
#tunggu sampe selesai
```

```

relax-santai@kelas-santai:~/myapps$ docker compose up -d --build
Compose can now delegate builds to bake for better performance.
To do so, set COMPOSE_BAKE=true.
[+] Building 2.5s (10/10) FINISHED
   Backend internal: load build definition from Dockerfile
   Backend internal: load .dockerignore
   Backend internal: load metadata for docker.io/library/golang:1.24.3-alpine
   Backend internal: load .dockerrignore
   Backend 1/4: File /tmp/docker.io/library/golang:1.24.3-alpine@sha256:b4f875e650466fa@fe62c6fd3f02517a392123eeaa85f1d7e699d85f780e4db1c1
   Backend 1/4: Build context
   Backend 1/4: Transfering context: 8MB
   Backend 1/4: Transferring context: 8MB
   Backend 1/4: WORKDIR /app
   Backend 1/4: COPY .
   Backend 1/4: RUN go mod download
   Backend 1/4: Exporting layers
   Backend 1/4: Writing image sha256:626298e8ac6972ebdf21baa57619ebc981d8676d27bb9748e68ab5b97711e4a
   Backend 1/4: Naming to docker.io/myapps-backend
   Backend resolving provenance for metadata file
[+] Building 0/0
  Backend          Built
  Network myapps_elatar-network Created
  Container mysql-container Started
  Container myapps-backend-1 Started
  Container phpmyadmin-container Started
kelas-santai@kelas-santai:~/myapps$ 
```

#cek phpmyadmin di port 8080

#cek servise myapp di 3000

Domain dan Subdomain

Domain adalah nama unik yang digunakan untuk mengakses suatu website di internet. Domain menggantikan penggunaan alamat IP server yang sulit diingat oleh manusia.

Contoh Domain:

- google.com
- tokopedia.com
- kelas-santai.com

Tanpa domain, pengguna harus mengetikkan alamat IP seperti:

<http://192.168.1.100>

Dengan domain, akses menjadi lebih praktis:

<http://kelas-santai.com>

Fungsi Domain

- Memberikan identitas untuk website di internet.
- Memudahkan pengunjung dalam mengingat alamat situs.
- Dapat digunakan sebagai brand, seperti tokopedia.com.
- Mewakili layanan tertentu, seperti:
 - Website utama
 - Sistem internal perusahaan
 - Aplikasi daring

Domain dapat dibeli dari layanan penyedia domain (domain registrar). Beberapa contoh populer:

- Niagahoster
- Domainesia
- Cloudflare
- Namecheap
- GoDaddy

Harga domain bervariasi tergantung dari ekstensi yang digunakan (.com, .id, .co, .tech, dll).

Apa Itu Subdomain?

Subdomain adalah bagian tambahan dari domain utama yang digunakan untuk memisahkan atau membedakan layanan tertentu dari website yang sama. Subdomain dituliskan di sebelah kiri dari domain utama.

Contoh Subdomain:

- `api.kelas-santai.com` → untuk layanan backend API
 - `blog.kelas-santai.com` → untuk halaman blog atau artikel
 - `admin.kelas-santai.com` → untuk dashboard admin
 - `cdn.kelas-santai.com` → untuk file statis (CDN)
-

Alasan Menggunakan Subdomain

Subdomain bermanfaat untuk membagi layanan berdasarkan fungsinya tanpa perlu membeli domain tambahan.

Kelebihan Subdomain:

Fitur	Penjelasan
Organisasi	Memisahkan sistem yang berbeda (API, admin, user, dll.)
Hemat Biaya	Cukup memiliki satu domain utama untuk membuat banyak subdomain
Fleksibilitas	Setiap subdomain dapat diarahkan ke server/port/aplikasi berbeda
Akses Mudah	Masing-masing layanan memiliki alamat sendiri yang mudah diakses

Contoh Penerapan Domain & Subdomain

Contoh struktur domain dan subdomain dalam sebuah proyek:

Alamat	Fungsi
<code>kelas-santai.com</code>	Website utama (frontend)
<code>api.kelas-santai.com</code>	Backend API (Node.js / Go)
<code>admin.kelas-santai.com</code>	Dashboard admin
<code>blog.kelas-santai.com</code>	Blog atau artikel

Setiap layanan bisa dijalankan di server dan port yang berbeda, namun tetap terlihat rapi dan terorganisir melalui struktur domain.

Cara Mengatur Subdomain di DNS

Pengaturan subdomain dilakukan melalui panel DNS pada layanan domain:

1. Tambahkan A Record:
 - Host: @ → mengarah ke IP server (untuk domain utama)

2. Tambahkan subdomain:
 - Host: api → mengarah ke IP server
 - Host: admin → mengarah ke IP server
 - Host: blog → mengarah ke IP server
-

Nginx dan Revers Proxy

NGINX (dibaca: "Engine X") adalah software web server yang ringan, cepat, dan sering digunakan untuk:

- Menyajikan konten website (web server)
- Mendistribusikan lalu lintas (load balancing)
- Mengarahkan permintaan ke server lain (reverse proxy)
- Menangani file statis (gambar, CSS, JS)
- SSL termination (HTTPS)

NGINX sangat populer di kalangan pengembang web dan DevOps karena performanya yang efisien dan kemampuannya menangani ribuan koneksi secara bersamaan.

Reverse Proxy adalah perantara antara client (misalnya browser pengguna) dan server backend. Saat client mengakses suatu alamat, reverse proxy menerima permintaan tersebut dan meneruskannya ke server lain di belakang layar.

Cara kerja:

1. Pengguna membuka <https://kelas-santai.com>
2. Permintaan masuk ke server NGINX
3. NGINX meneruskan permintaan ke server backend di port lain (misalnya port 3000)
4. NGINX menerima respons dari backend dan mengirimkannya kembali ke pengguna

Pengguna tidak tahu bahwa aplikasi backend sebenarnya berjalan di port lain seperti `localhost:3000` — pengguna hanya melihat domain utamanya saja.

Reverse proxy memberikan banyak manfaat:

Manfaat	Penjelasan
Keamanan	Backend tidak langsung diakses dari luar
Organisasi	Bisa mengatur banyak layanan di satu domain dengan subdomain atau path
Load Balancing	Dapat membagi permintaan ke beberapa server backend
SSL Termination	NGINX menangani HTTPS, backend tetap berjalan dengan HTTP biasa
Cache & Kompresi	NGINX dapat meng-cache konten atau mengompres data untuk efisiensi
Custom Error Page	Menampilkan halaman error sendiri bila backend mati atau lambat

Misalnya ada beberapa aplikasi seperti ini:

Alamat	Fungsi	Backend
kelas-santai.com	Website utama (frontend)	localhost:3000
api.kelas-santai.com	REST API (backend Go/Node.js)	localhost:8000
admin.kelas-santai.com	Admin Panel	localhost:8080

Semua alamat itu diarahkan oleh NGINX ke server yang berbeda.

Contoh Konfigurasi NGINX (Reverse Proxy dengan Subdomain)

```
server {
    listen 80;
    server_name kelas-santai.com;

    location / {
        proxy_pass http://localhost:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}

server {
    listen 80;
    server_name api.kelas-santai.com;

    location / {
        proxy_pass http://localhost:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}

server {
    listen 80;
    server_name admin.kelas-santai.com;

    location / {
        proxy_pass http://localhost:8080;
    }
}
```

```
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
}
}
```

Reverse Proxy + HTTPS

Agar lebih aman, reverse proxy bisa dikombinasikan dengan SSL menggunakan **Let's Encrypt** (sertifikat gratis). Contoh menggunakan [Certbot](#) untuk mengaktifkan HTTPS:

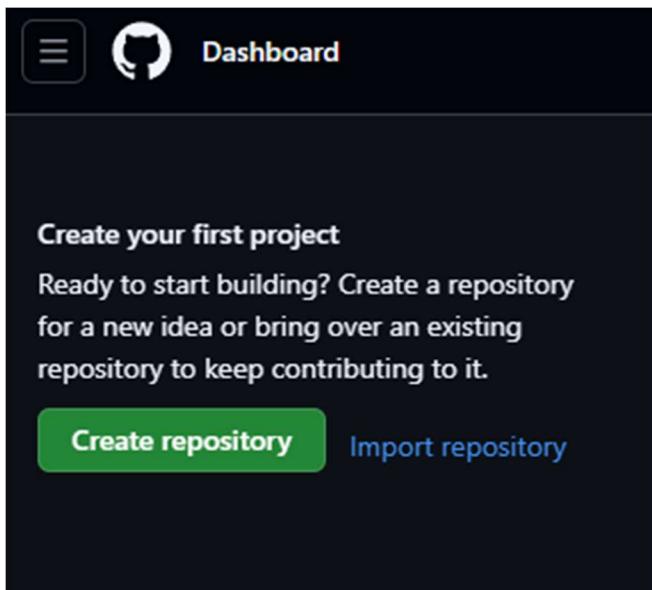
```
sudo apt install certbot python3-certbot-nginx
sudo certbot --nginx
```

GITHUB WEBHOOK

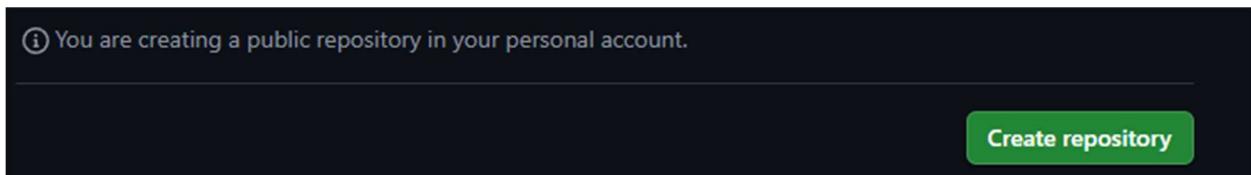
GitHub adalah sebuah platform hosting berbasis web yang menyediakan layanan penting untuk versi kontrol dan kolaborasi pengembangan kode menggunakan sistem Git. Dengan GitHub, banyak developer dari berbagai tempat dapat bekerja secara bersama-sama dalam satu proyek perangkat lunak dengan lebih mudah dan terorganisir.

Platform ini memungkinkan pengelolaan perubahan kode secara efisien, sehingga setiap perubahan yang dibuat dapat dilacak dengan jelas melalui riwayat pengembangan. Hal ini sangat membantu dalam menjaga kualitas kode, menghindari konflik, dan memudahkan kolaborasi tim dalam skala kecil maupun besar. Untuk memulai menggunakan GitHub, kunjungi situs resmi di <https://github.com>. Di sana, dapat membuat akun baru secara gratis dan langsung login untuk mulai mengelola proyek, membuat repository, serta berkolaborasi dengan developer lain di seluruh dunia.

Membuat Repository Baru



Untuk pengguna baru, pembuatan repository bisa klik "Create repository" lalu isi Repository Name, buat repository itu public/private. Lalu create repository



Menambahkan kode ke dalam Repository GitHub

dalam folder project kita yang terdiri dari beberapa file yaitu adalah go.mod, main.go, dan Dockerfile. di direktori tersebut kita bisa menambahkan untuk mengupload kode ke dalam github . Dalam beberapa versi untuk versi terbaru github tidak bisa untuk menclone repository git dari dalam server, jadi harus login terlebih dahulu ke dalam server agar bisa mengakses git, namun hal ini bisa dilakukan dengan memakai kode token akses dari github

Token Akses Github

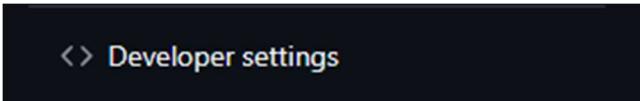
Untuk mendapatkan kode token akses yaitu klik logo profile di pojok kanan atas,



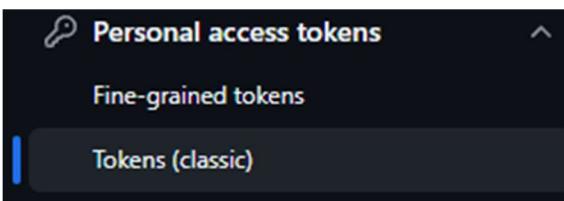
Cari menu Setting



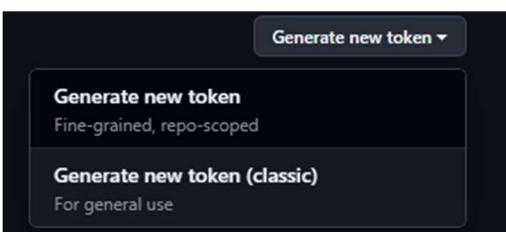
Cari menu developer setting



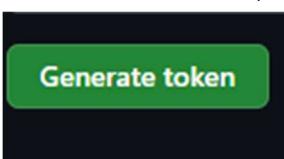
Pilih Personal Access tokens



Pilih Generate new token dan Generate new Token (classic)



Berikan nama token, waktu exp dan centang semua bagian scope dan generate token



Simpan kode akses yang aman di word, txt atau yang lainnya

Berikut contoh dari persola token akses

ghp_15gcPb6rFEfg2vGHXUbf0QM5QLZYN613SSXS

#Copy link repository git hub

<https://github.com/kelas-santai/backend-service.git>

lalu ubah menjadi format nya seperti ini :

<https://tokenakses@repositorygithub>

contoh:

https://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@github.com/kelas-santai/backend-service.git

Menambahkan code ke repository

```
...or create a new repository on the command line
echo "# backend-service" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/kelas-santai/backend-service.git
git push -u origin main
```

pada repository git akan terdapat otomatis untuk membuat repository pada project,

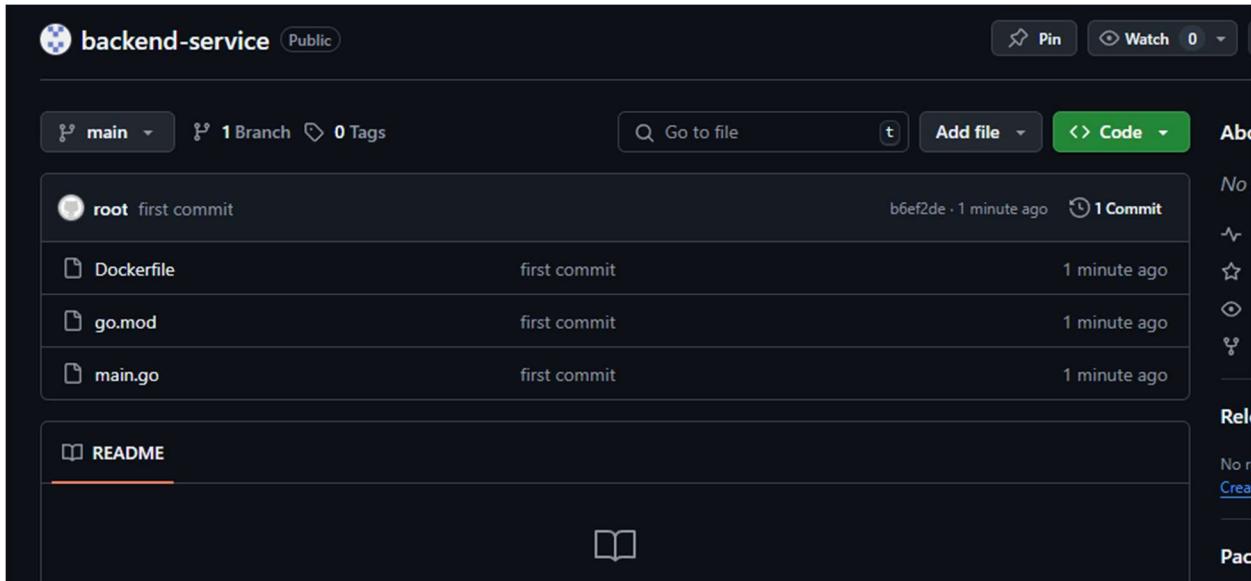
#untuk menambahkan semua file yang di masukan pakai perintah

git add .

\$ganti url yang di perintah "git remote add origin " menjadi menggunakan token akses yang berarti
git remote add origin

https://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@github.com/kelas-santai/backend-service.git

```
kelas-santai@kelas-santai:~/myapps/myapp$ sudo git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 742 bytes | 742.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/kelas-santai/backend-service.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
kelas-santai@kelas-santai:~/myapps/myapp$ |
```



Menclone Repository

Untuk mengclone serpository dari git hub perintah dari git yaitu adalah “`sudo git clone https://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@github.com/kelas-santai/backend-service.git`“

WEBHOOK

Webhook adalah mekanisme komunikasi otomatis berbasis HTTP yang digunakan untuk mengirimkan data secara real-time dari satu sistem ke sistem lainnya, ketika sebuah peristiwa (event) terjadi. Alih-alih harus terus menerus memeriksa (polling) apakah ada perubahan, webhook memungkinkan sistem penerima langsung mendapatkan notifikasi secara instan begitu event terjadi.

Cara Kerja Webhook

1. Sistem sumber (misalnya GitHub, Stripe, atau aplikasi internal) mengirimkan permintaan **HTTP POST** ke sebuah **endpoint URL** yang sudah ditentukan.
2. Data dikirim dalam format seperti JSON, berisi informasi tentang event yang terjadi.
3. Sistem penerima membaca data tersebut dan melakukan aksi sesuai logika yang telah dibuat (misalnya deploy aplikasi, mengirim notifikasi, menyimpan data, dll).

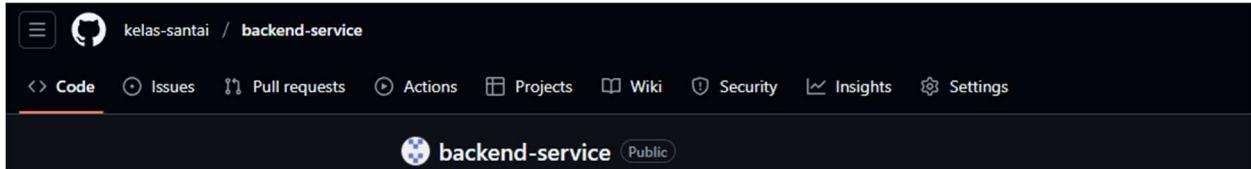
Contoh Penggunaan Webhook

- **GitHub Webhook:** Mengirim notifikasi ke server saat terjadi push ke repository, untuk memicu build atau deploy otomatis.
- **Payment Gateway Webhook (misal Midtrans, Stripe):** Memberi tahu server bahwa pembayaran telah berhasil.
- **Form Webhook (misal Google Forms + webhook):** Mengirim data form ke API secara langsung.

Membuat Webhook Github dengan service python

Setting Github Webhook

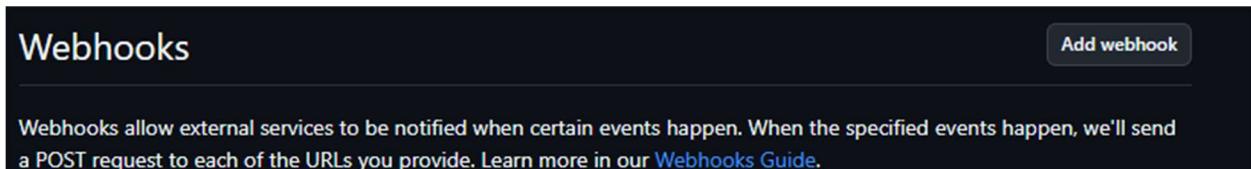
#pilih setting pada menu repository di atas



Pilih menu bagian webhook



buat webhook baru



#Isi pengaturan seperti gambar di bawah ini

Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *
https://example.com/postreceive

Content type *
application/json

Secret
webhook

SSL verification
 By default, we verify SSL certificates when delivering payloads.
 Enable SSL verification Disable (not recommended)

Which events would you like to trigger this webhook?

Just the push event.
 Send me everything.
 Let me select individual events.

Active
We will deliver event details when this hook is triggered.

Add webhook

Note :

Payload URL di isi link API Endpoint yang di arahkan ke server

Contoh :

<https://baseurl/backend-servise-bagja>

