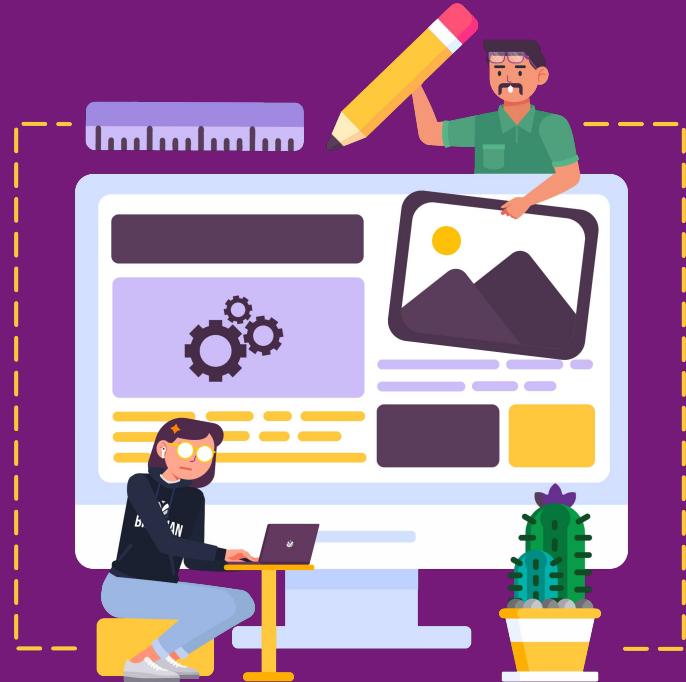


# From Command Line to Collaboration: Terminal, IDE, and GIT

**Silver** Chapter 1 - Topic 2

Selamat datang di **Chapter 1 Topic 2**  
online course **Fullstack Web** dari BINAR!

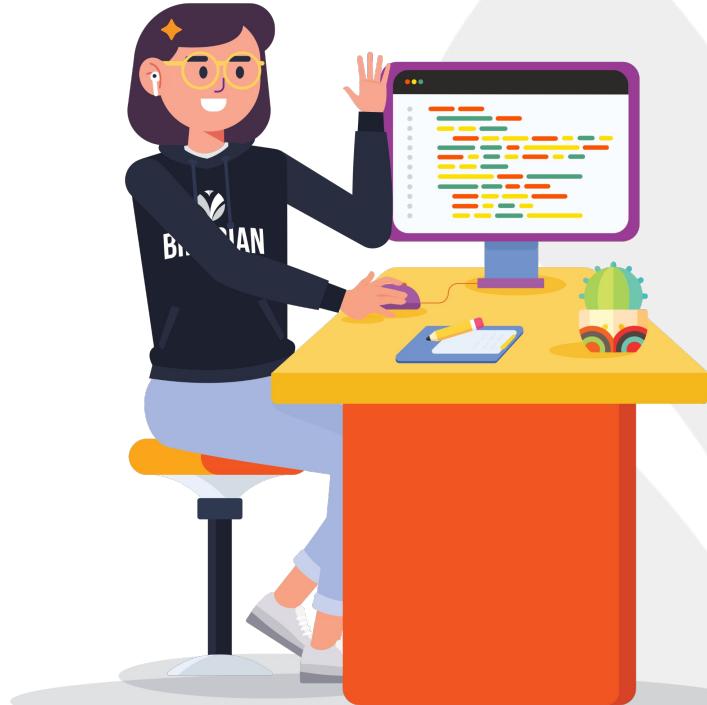


### Mari berpetualang kembali!🌟

Pada topic sebelumnya, kita udah membahas mengenai Digital Product Development. Nah, di topic 2 ini kita akan kupas tuntas tools dalam proses pengembangan web agar menjadi mudah, yaitu **Terminal dan IDE**.

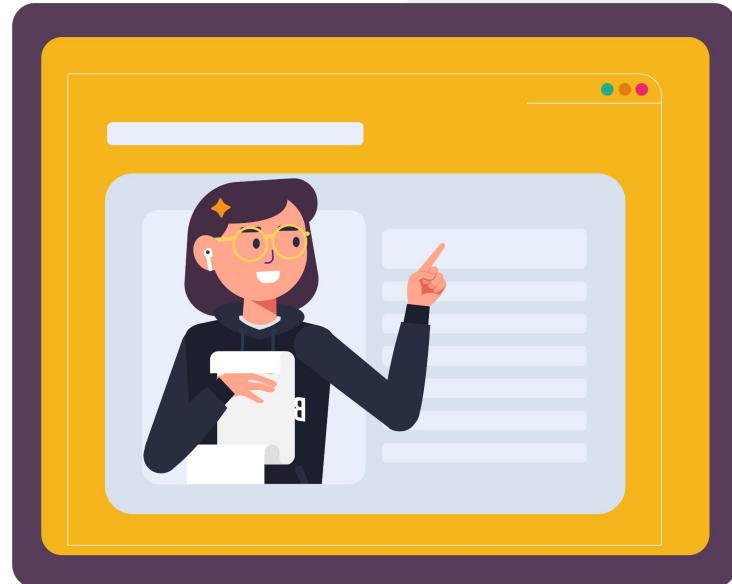
Nggak cuma itu, kita juga bakal menelusuri tentang **GIT**, alias Google Drive-nya Programmer yang berfungsi agar Developer mudah berkolaborasi dengan Developer lain.

Let's go kita cari tau bareng~



## Detailnya, kita bakal bahas hal-hal berikut ini:

- Pengantar Terminal
- Konsep Perintah dan Argument
- Perintah yang umum dipakai di Terminal
- IDE dan cara penerapannya
- Konsep Git sebagai version control
- Cara penggunaan Git
- Instalasi, inisialisasi, dan commit GIT
- Penggunaan Git di Remote Repository
- Konsep Branching untuk berkolaborasi
- Commit behind untuk menyelesaikan konflik



**Topic ini difokuskan pada acceptance criteria berikut ini:**

**Mampu menggunakan Git sebagai version control**

Mampu membuat halaman web sesuai dengan desain yang diberikan dengan menggunakan HTML dan CSS dan memastikan penulisannya terstruktur dan rapi

Mampu membuat website pada poin 2 bersifat responsive pada berbagai jenis layar dengan menggunakan CSS Bootstrap

Mampu membuat website sesuai dengan desain yang ada di Figma

Terminal?

Hmmm... yang bakal kita bahas bukan terminal yang banyak bus itu ya.

Bukan juga aplikasi yang dipakai hacker-hacker di film-film hollywood gitu.

Cekidoooot kita langsung aja ya~



### Pernah kepikiran nggak?

Kenapa kalau kita mengetik di keyboard, hurufnya bisa langsung muncul di layar?

Yap! Semua yang terjadi di layar karena ada terminal di belakangnya, gengs.

Jadi, terminal itulah yang menerima input dari keyboard dan menampilkan outputnya di layar (screen).



### Mengembangkan website itu bisa pakai terminal lho ternyata~

Bener sob, pas kita sedang mengembangkan sebuah website, kita pastinya butuh menjalankan website tersebut di laptop atau komputer kita masing-masing untuk:

- Memastikan apakah kode kita berjalan dengan baik
- Memastikan tampilan website kita sesuai dengan desainnya



Salah satu cara paling efektif buat menjalankan website kita adalah dengan menggunakan terminal.

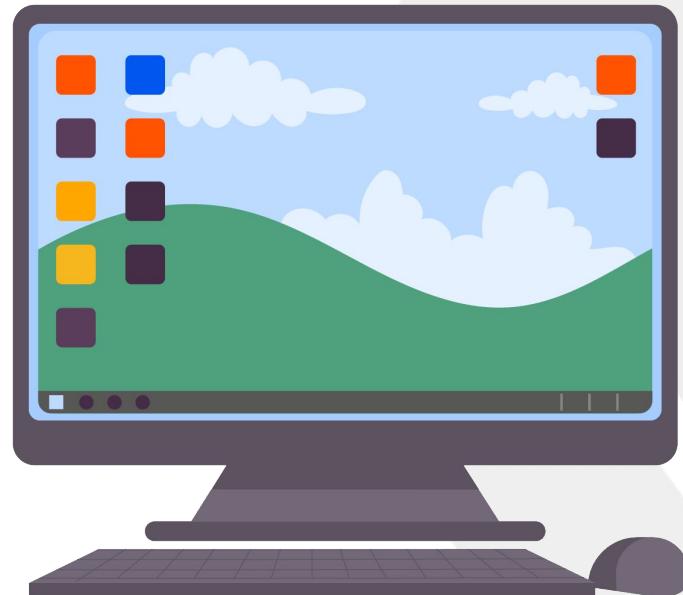
Dengan terminal, kita bisa **mengetik perintah-perintah canggih buat nyuruh-nyuruh laptop atau komputer kita tanpa perlu GUI**.



### GUI itu apa sih? 🤔

GUI itu contohnya kayak ikon yang biasa kita lihat di PC atau handphone saat membuka aplikasi.

Nah, dalam proses pembuatan website ataupun aplikasi, **nggak semua tools yang digunakan sama developer ada GUI-nya gengs.**

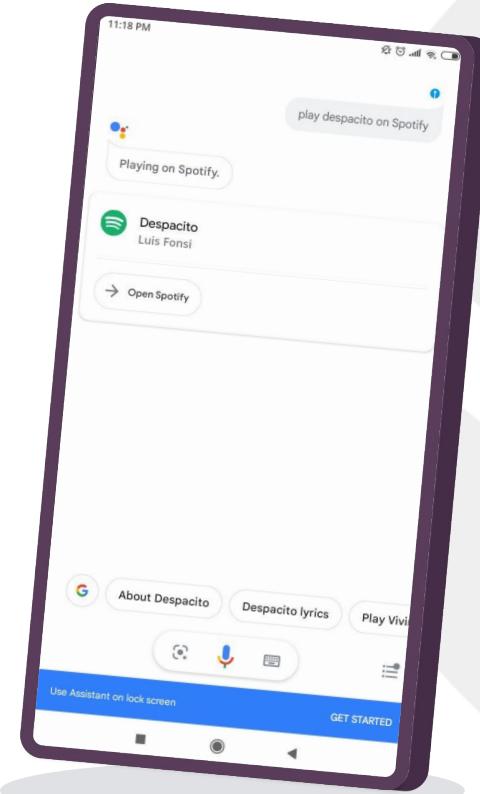


### Kamu pernah pakai Google Assistant, Siri, atau Alexa kan?

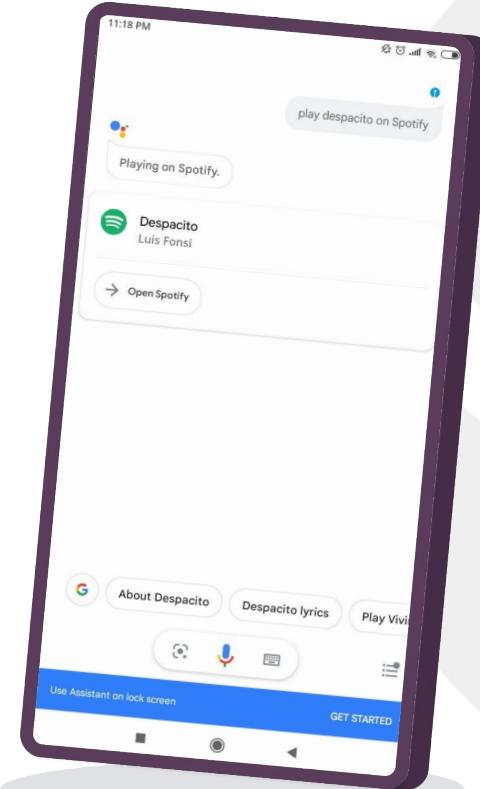
Google Assistant, Siri, atau Alexa ini cara kerjanya mirip terminal. Pas kita mengetik atau menyebutkan sesuatu ke mereka, mereka bakal melakukan sesuai perintah kita.

Contohnya kaya gambar di samping.

Kita minta Google Assistant buat menyalakan lagu Despacito di Spotify.



Kalau kamu perhatikan, disana kita memasukkan input berupa teks, yaitu “**play** despacito on Spotify”, nantinya Google Assistant langsung **menginterpretasikan perintah tersebut dan menjalankannya dengan membuka aplikasi Spotify dan memainkan lagu Despacito.**



Cara kerja Terminal mirip dengan Google Assistant yang tadi sudah kita bahas.

Bedanya, kalau Google Assistant bisa terima perintah dalam bentuk suara, kalau **terminal cuma menerima perintah dalam bentuk teks aja**.



### Terus, contoh aplikasi terminal itu ada apa aja ya? 🤔

Buanyak! Hehehe~ Bahkan ada yang spesifik untuk OS yang kamu pakai, lho!

- Pengguna Windows bisa menggunakan aplikasi terminal emulator bernama **Powershell**.
- Kalau kamu pakai MacOS, ada aplikasi terminal emulator bernama **iTerm**.
- Kalau pakai GNU/Linux gimana? Kamu bisa pakai aplikasi terminal emulator bawaan, biasanya **xterm**.



**Kalau terminal ada banyak, cara pakainya beda nggak? Terus, paling baik belajar yang mana?**

Tenang aja, meskipun terminal banyak jenisnya, cara pakainya itu sama aja.

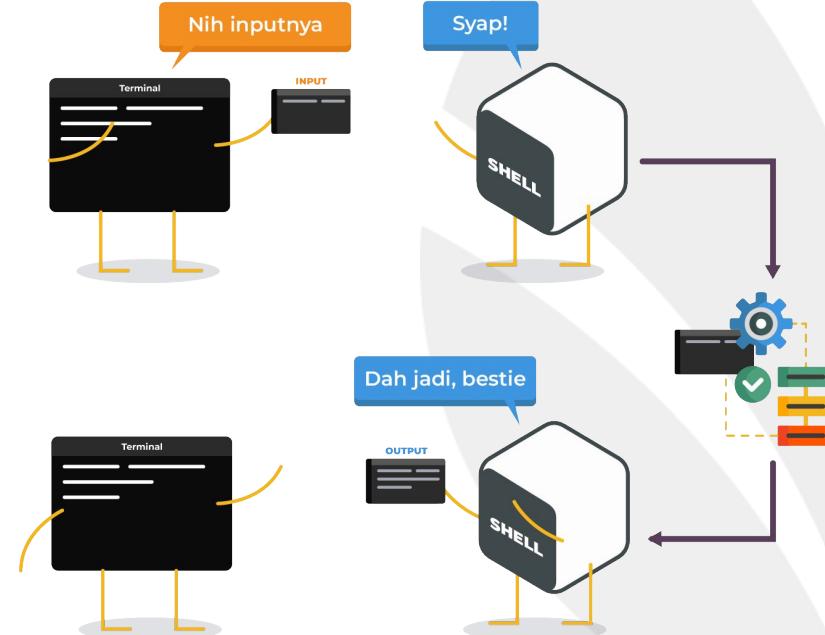
Soalnya, yang bakalan **kita ketik di dalam sebuah terminal itu adalah sebuah teks yang disebut sebagai shell**.



Shell adalah program yang memproses input dan menjalankan perintah. Shell ini diakses lewat terminal.

Shell bekerja dengan menerima input dari keyboard, lalu menjalankannya sebagai perintah (command), dan menampilkan hasilnya di layar.

Karena input yang diterima itu dari keyboard, maka sebagian fungsi dari shell ini berupa teks.



Shell ini sebenarnya ada banyak, cuma kita bisa pisahkan jadi 2 jenis aja, yaitu:

- **Unix-like shell (zsh, dan bash) (Linux, MacOS, WSL).**
- **Non Unix-like shell (Powershell, Windows).**

Kedua jenis shell tadi dibuat dengan prinsip yang sama dan cara pakainya juga mirip.

**Di topic ini kita bakalan pakai dan bahas jenis Unix-like shell aja, karena mayoritas server yang banyak digunakan berbasis Linux.**

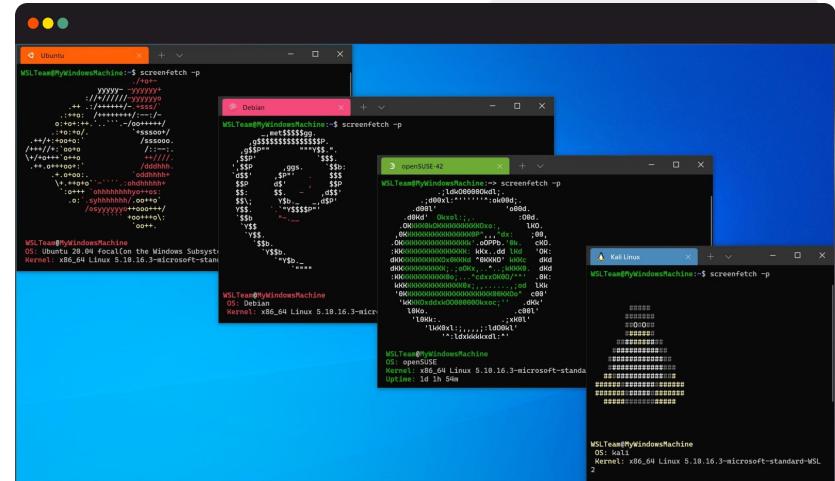


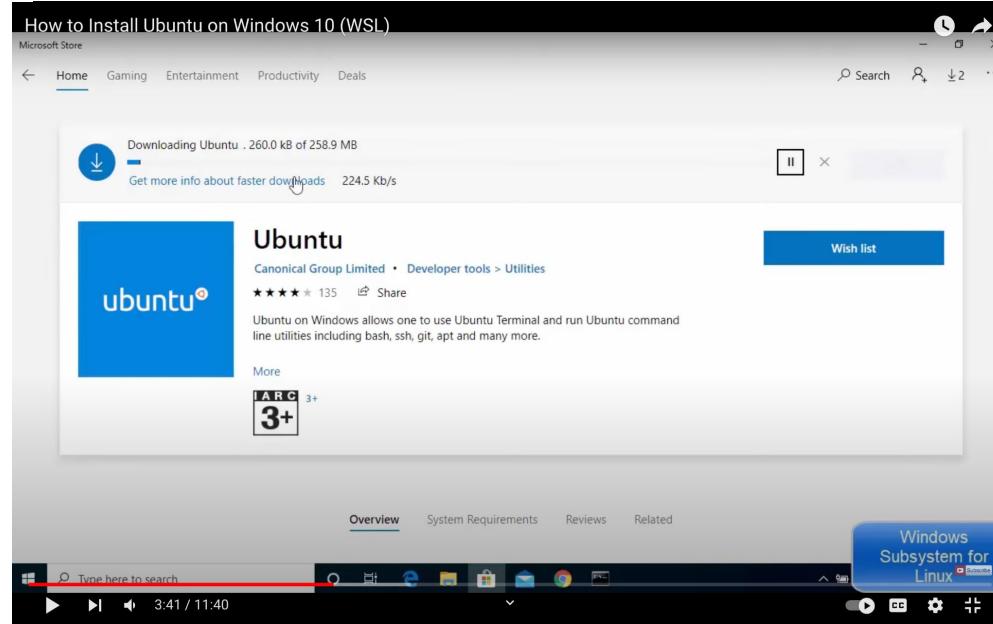
# Kalau aku pakai Windows, terus gimana dong?

Santuyyy~

Bill Gates sekarang lagi dekat-dekatnya nih sama Linux, jadi sekarang di Windows udah ada fitur namanya WSL (Windows Subsystem for Linux).

Dengan menggunakan WSL, kita bisa pakai Terminal Linux di Windows. Sehingga kamu nggak perlu ganti OS buat pakai Terminal yang ada di Linux dan MacOS.





Untuk instalasi WSL, bisa kamu lihat di link berikut [How to Install Ubuntu on Windows 10 \(WSL\)](#)

Sangat disarankan untuk install distribusi Ubuntu.

### Setelah instalasi, kita coba aplikasi terminal emulatornya yuk!

- Buat Windows User, silahkan buka aplikasi bernama **Ubuntu** setelah kamu selesai melakukan instalasi WSL di laptop atau komputer kamu.
- Buat MacOS user, silahkan buka aplikasi bernama iTerm.
- Dan buat GNU/Linux user, kamu pasti tahu harus buka apa, biasanya bisa dibuka pakai shortcut **CTRL + SHIFT + ENTER**.



### Kita coba langkah pertama untuk tulis perintah di terminal yuk~

Kita coba menulis perintah pertama yang berguna untuk memunculkan suatu teks di dalam terminal. Kamu bisa coba tulis perintah ini di dalam terminal dan tekan **enter**.

```
echo "Hello World"
```

Output yang akan muncul di dalam terminal nanti bakal terlihat kayak gambar di samping.

A screenshot of a dark-themed terminal window on a Mac OS X desktop. The window has three colored circular icons in the top-left corner. The text area contains the command "echo "Hello World"" followed by its output "Hello World" and a cursor at the end of the line.

```
λ ~/ echo "Hello World"
Hello World
λ ~/ |
```

Sekarang kamu udah belajar tentang terminal. Yap, ini adalah langkah awal untuk kamu bisa mengembangkan sebuah website. Perdalam lagi prakteknya agar kamu terbiasa dalam menggunakan.



Pas menulis di dalam terminal, kita bakal sering bersinggungan sama yang namanya **perintah dan argumen**.

Eits, tapi ini juga bukan argumen buat debat gitu yaa.

Biar nggak salah paham, letsgow kita langsung pelajari~



### So, Perintah dan Argumen itu apa dong?

Kita pakai contoh perintah “**play Glimpse of Us on Spotify**” tadi. Perintah yang kita tulis dalam contoh tersebut bisa kita jabarkan menjadi dua, yaitu **perintah dan argumen**.

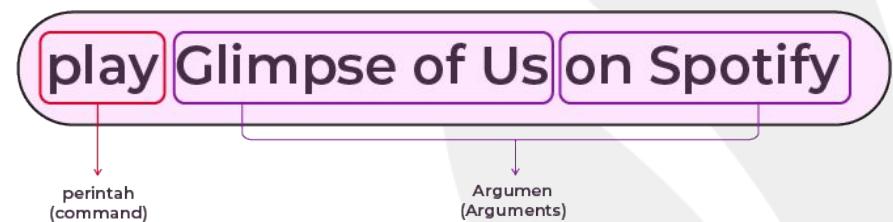
- **Perintah** ini biasanya adalah sebuah aktivitas yang ada di dalam komputer kita.
- **Argumen** adalah data yang kita berikan kepada perintah atau aplikasi tersebut.



### Elaborasi dari contoh tadi gimana ya?

Begini sob, tadi kita memerintahkan “play Glimpse of Us on Spotify” yekan? Nah, yang dimaksud **perintah adalah play**, di mana ini dalam komputer kita ada sebuah program bernama play.

Sedangkan **argumen dari perintah tersebut adalah Glimpse of Us on Spotify**, yang mana data ini akan diolah sama perintah terkait.



Kalau kita ambil contoh echo “Hello World”, sebenarnya kita sedang memanggil perintah atau aplikasi yang bernama **echo**, dengan argumen **Hello World** di mana bakal jadi output di dalam terminal.

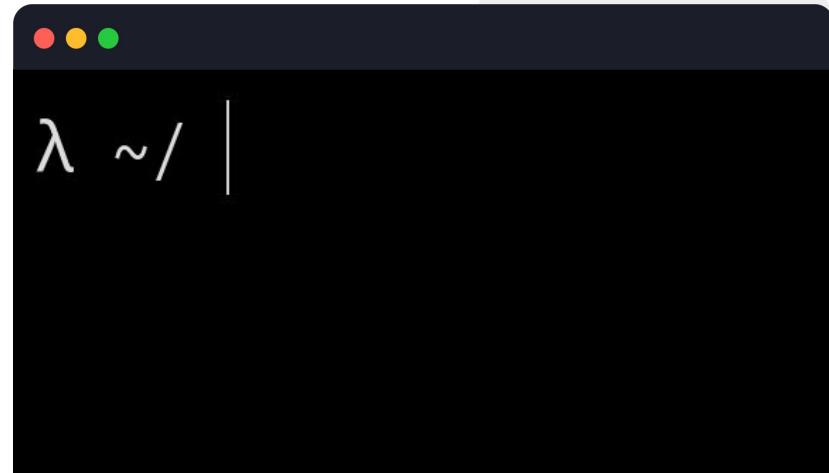
Gimana? Sampai sini udah ada bayangan?



### Navigasi dalam Terminal~

Pas kita membuka aplikasi terminal, kita bakal masuk ke dalam sebuah direktori di dalam komputer kita. Direktori tersebut adalah direktori user, yang biasa dilambangkan dengan menggunakan **tilde (~)**.

Di terminal, kita juga bisa melakukan navigasi buat pindah ke direktori lain lho. Caranya gimana ya?



### Sebelum belajar gimana caranya navigasi di dalam terminal, kita perlu tahu Path dulu nih!

Path adalah sebuah **alamat dari suatu file atau direktori di dalam komputer** kita.

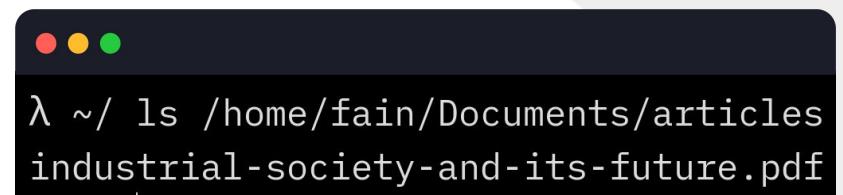
- kalau di Windows, path itu biasanya kayak gini:  
**C:\Program Files\Tetris\Tetris.exe**
- Tapi kalau di sistem operasi yang mirip dengan Unix, biasanya kayak gini:  
**/home/fain/Pictures/Tetris.png**

Nantinya kita bakal lebih sering menggunakan versi Unix~



**Path sendiri ada dua jenis, yaitu absolute path dan relative path**

1. **Absolute Path:** adalah sebuah path yang ditulis mulai dari root directory atau slash. Contoh:  
**/home/sabrina/Pictures/Tetris.png**

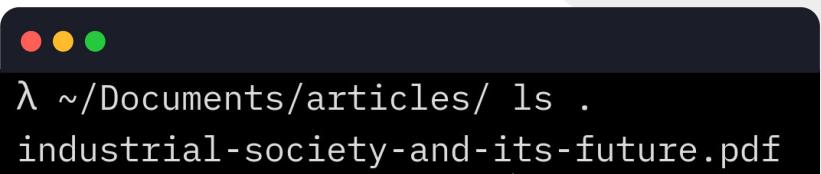


```
λ ~/ ls /home/fain/Documents/articles  
industrial-society-and-its-future.pdf
```

1. **Absolute Path:** adalah path yang menggunakan direktori root sebagai referensi.
2. **Relative Path:** adalah path yang digunakan dengan menggunakan direktori yang sedang dibuka sebagai referensi.

Sebagai contoh, sekarang kita berada di dalam direktori Pictures, maka kita bisa memanggil file Tetris.png secara relatif.

**./Tetris.png**



```
λ ~/Documents/articles/ ls .
industrial-society-and-its-future.pdf
```

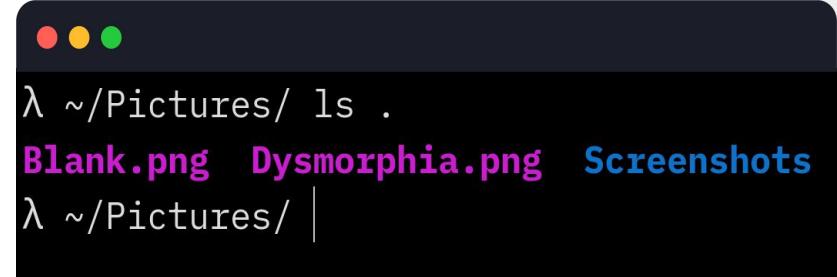
A screenshot of a terminal window with a dark background and light-colored text. At the top, there are three small colored circles (red, yellow, green) representing window control buttons. Below them, the text 'λ' followed by the command '~/Documents/articles/ ls .' and the resulting output 'industrial-society-and-its-future.pdf' are displayed.

Oiya, ada dua jenis teks atau simbol yang bisa kita pakai pas mau menulis sebuah **relative path**.

## 1. Current Directory (.)

Simbol satu titik digunakan buat memanggil direktori yang lagi kita buka.

Misalnya, sekarang kita lagi ada di dalam direktori **Pictures**. Kalau kita menggunakan perintah **ls** dan diikuti dengan **titik**, artinya kita lagi menyuruh komputer buat menampilkan file atau direktori apa aja yang ada di dalam direktori saat ini.



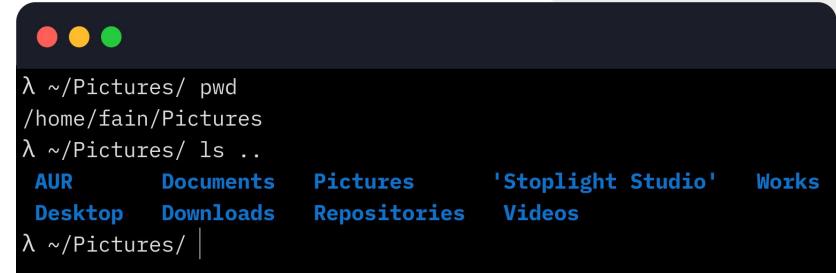
```
λ ~/Pictures/ ls .
Blank.png Dysmorphia.png Screenshots
λ ~/Pictures/ |
```

### 2. Parent Directory (..)

Simbol dua titik **digunakan buat memanggil parent directory dari directory saat ini.**

Misalnya, kita lagi ada di dalam path  
**/home/sabrina/Pictures**

Artinya, sabrina adalah parent directory dari direktori Pictures. Kita bisa gunakan perintah ls dan diikuti sama argumen dua titik untuk menampilkan apa aja yang ada di dalam direktori sabrina.



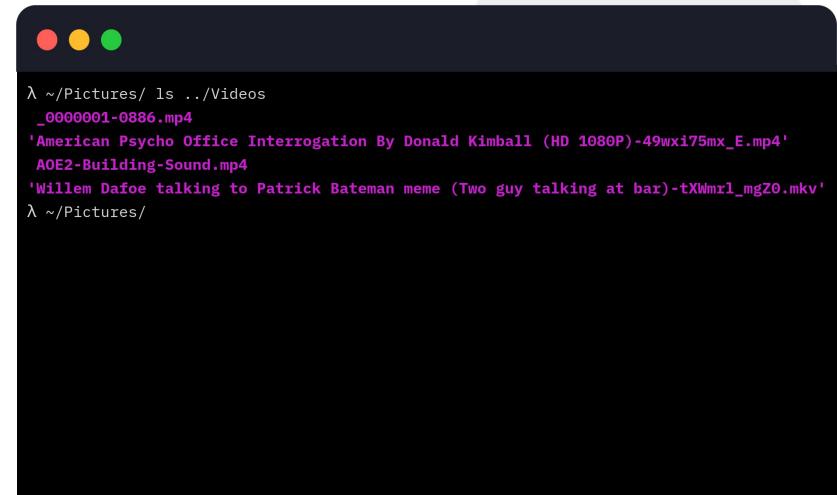
```
λ ~/Pictures/ pwd
/home/fain/Pictures
λ ~/Pictures/ ls ..
AUR      Documents    Pictures      'Stoplight Studio'   Works
Desktop  Downloads    Repositories  Videos
λ ~/Pictures/ |
```

Relative path ini bisa diteruskan path-nya. Misal, pas kita lagi ada di dalam path

### **/home/sabrina/Pictures**

Tapi kita mau melihat apa isi dari directory Download, yang ada di path berikut:

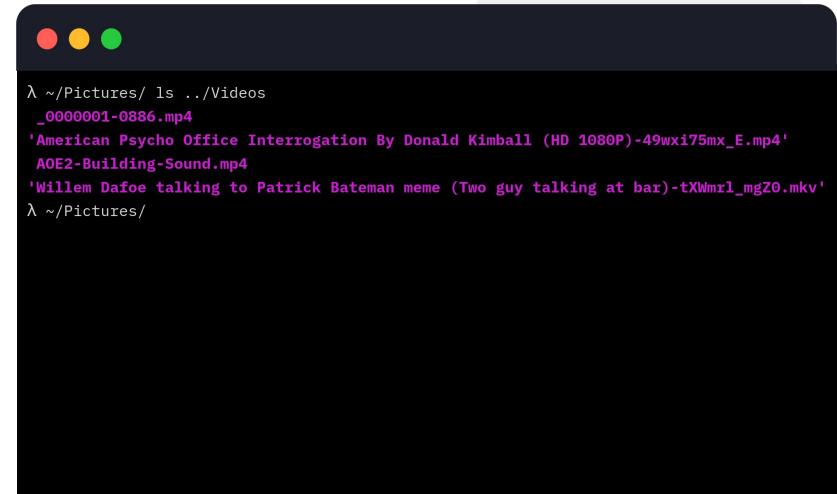
### **/home/sabrina/Videos**

A screenshot of a terminal window with a dark background. It shows the command 'ls .. /Videos' being run from the directory '/home/sabrina/Pictures'. The output lists three video files: '\_0000001-0886.mp4', 'American Psycho Office Interrogation By Donald Kimball (HD 1080P)-49wxij75mx\_E.mp4', and 'AOE2-Building-Sound.mp4'. Below these, another line of code is partially visible: 'Willem Dafoe talking to Patrick Bateman meme (Two guy talking at bar)-tXWmrxl\_mgZ0.mkv'. The terminal window has a standard OS X-style title bar with red, yellow, and green buttons.

Maka kita bisa menuliskan path seperti di bawah ini:

### **.. / Videos**

Path directory yang saat ini baru dibuka di terminal  
biasa disebut sebagai Current Working Directory (CWD).  
Nah relative path ini adalah path yang relatif terhadap  
CWD tadi.

A screenshot of a macOS terminal window. The title bar shows three colored circles (red, yellow, green). The command entered is `λ ~/Pictures/ ls .. /Videos` followed by several video file names: `\_0000001-0886.mp4`, "'American Psycho Office Interrogation By Donald Kimball (HD 1080P)-49wxij75mx\_E.mp4'", "AOE2-Building-Sound.mp4", and "'Willem Dafoe talking to Patrick Bateman meme (Two guy talking at bar)-tXWmrxl\_mgZ0.mkv'". The prompt ends with `λ ~/Pictures/`.

```
λ ~/Pictures/ ls .. /Videos
_0000001-0886.mp4
'American Psycho Office Interrogation By Donald Kimball (HD 1080P)-49wxij75mx_E.mp4'
AOE2-Building-Sound.mp4
'Willem Dafoe talking to Patrick Bateman meme (Two guy talking at bar)-tXWmrxl_mgZ0.mkv'
λ ~/Pictures/
```

### Change Directory!

Nah, setelah kamu tau apa itu path, sekarang saatnya kita cobain perintah yang digunakan untuk berpindah-pindah direktori di dalam terminal, selayaknya kita menggunakan File Explorer.

Buat **berpindah direktori di dalam terminal, kita bisa menggunakan perintah bernama cd**. Di mana perintah cd ini meminta path sebagai argumen, baik itu relative path maupun absolute path.

Perintah cd ini bakal mengubah current working directory.

```
λ ~/ cd Downloads  
λ ~/Downloads/ cd ../Documents  
λ ~/Documents/ cd /home/fain.Repositories  
λ ~/Repositories/
```

Perintah dan argumen ini ternyata berfungsi untuk mengubah directory juga ya. Sekarang jadi tau kan gimana mengubah directory di dalam terminal? Tapi apakah fungsinya hanya untuk pindah directory aja? 😕



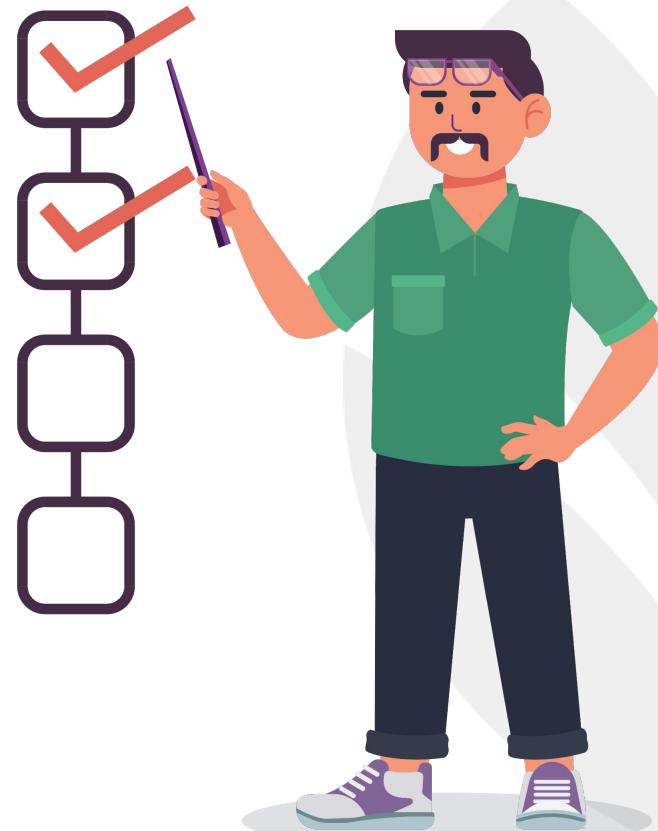
Eits, gak kok... fungsinya ga hanya pindah direktori aja. Ada **perintah-perintah di terminal** lainnya yang akan kita bahas. Dan pastinya bakal berguna banget. Cekidot~



**Berikut perintah yang umum  
dipakai terkait dengan terminal~**

1. ls
2. cat
3. touch
4. rm
5. mkdir
6. echo
7. mv path/ new/path
8. top
9. ps -aux
10. chmod

Kita cek satu persatu, yuk!



## 1. ls

ls merupakan **perintah terminal yang berguna buat menampilkan konten dari sebuah directory**. Sama kayak cd, perintah ini meminta **path** sebagai argumen.



A screenshot of a macOS-style terminal window. The title bar shows three colored window control buttons (red, yellow, green). The main area of the terminal displays the output of the 'ls' command run from the directory '~/Repositories/'. The output lists several directory names: bookstoresvc, buf, coffeesvc, dotfiles, express-public, fikrirnurhidayat, gatsby-starter-elements, GO, grpc-gateway-boilerplate, next-elements, serat, stoplight-gatsby, stoplight-next, and stoplight-react. The terminal prompt at the bottom is 'λ ~/Repositories/'.

```
λ ~/Repositories/ ls .
bookstoresvc           GO
buf                   grpc-gateway-boilerplate
coffeesvc             next-elements
dotfiles              serat
express-public        stoplight-gatsby
fikrirnurhidayat     stoplight-next
gatsby-starter-elements stoplight-react
λ ~/Repositories/
```

## 2. cat

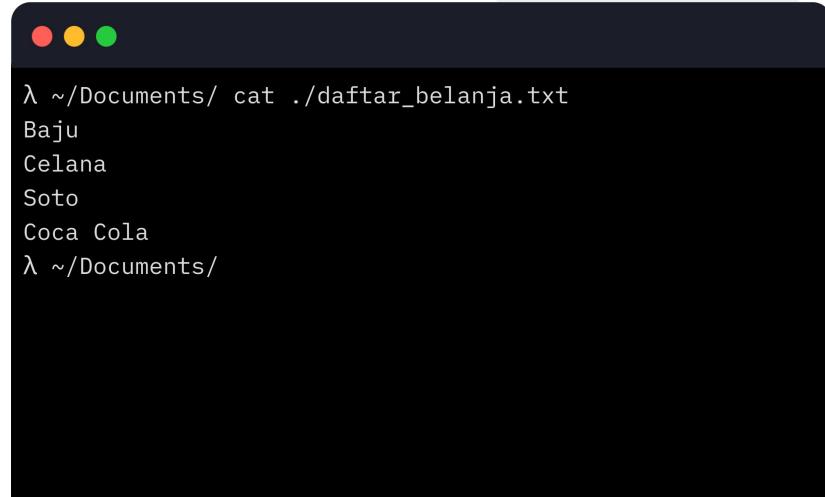
Perintah ini bukan perintah buat ngelus-ngelus kucing ya! 😺

Perintah ini dipakai buat **menampilkan konten dari sebuah file**.

Misal, kalian punya daftar belanja yang kalian simpan di sebuah file bernama

**daftar\_belanja.txt**. Nah, kalian bisa melihat isi dari file tersebut melalui terminal dengan menggunakan perintah cat.

Sama seperti ls dan cd, cat meminta path sebagai argumen, namun path tersebut haruslah merujuk ke sebuah file.

A screenshot of a dark-themed terminal window. At the top, there are three colored window control buttons (red, yellow, green). Below them, the command `λ ~/Documents/ cat ./daftar\_belanja.txt` is entered. The terminal then displays the contents of the file: "Baju", "Celana", "Soto", "Coca Cola". Finally, it shows the prompt again: `λ ~/Documents/`. The background of the slide features a large, light gray graphic of a leaf or petal shape.

### 3. touch

Perintah ini bisa kita pakai untuk **membuat sebuah file di dalam terminal**. Perintah ini meminta path juga sebagai argumen.

Biasanya perintah ini dikombinasikan dengan echo, yang digunakan untuk memasukkan teks ke dalam file tersebut.

You know, di dalam Unix-like system, semua hal itu adalah file.

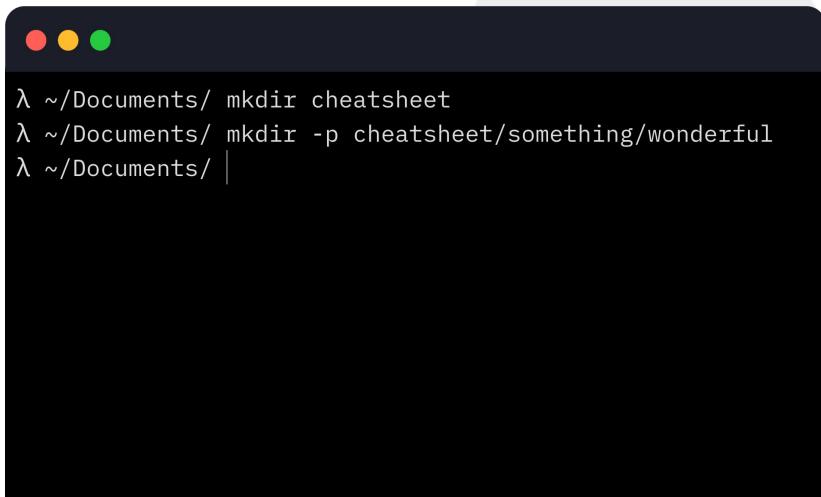
```
λ ~/Documents/ touch ./tujuan_hidup.txt
λ ~/Documents/ echo "Gaada" >> ./tujuan_hidup.txt
λ ~/Documents/ cat ./tujuan_hidup.txt
Gaada
λ ~/Documents/
```

## 4. **mkdir**

Perintah ini **digunakan untuk membuat direktori**.

Mirip dengan cd, perintah ini meminta path sebagai argumen.

Apabila kita mau membuat direktori yang bersarang, kita bisa menambahkan flag **-p** supaya semua direktori yang belum ada pada path-nya bakal otomatis dibuat..



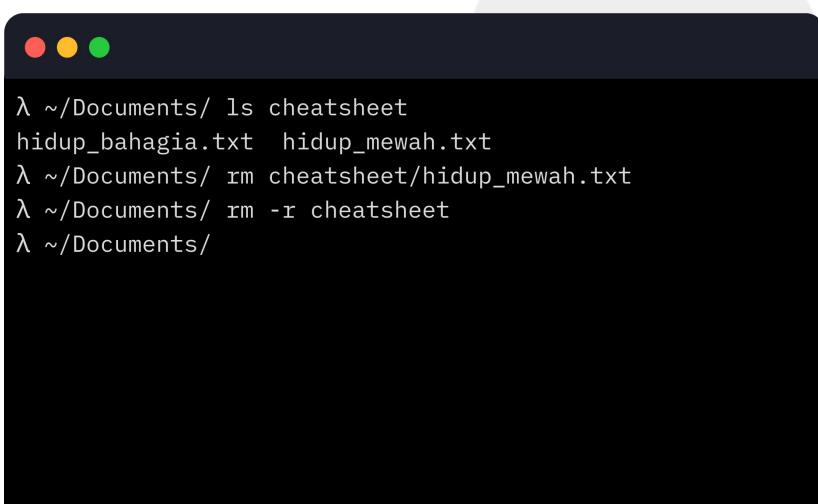
```
λ ~/Documents/ mkdir cheatsheet
λ ~/Documents/ mkdir -p cheatsheet/something/wonderful
λ ~/Documents/ |
```

### 5. rm

Kalau ada yang dibuat, pasti ada yang dihapus juga.

Nah, perintah rm ini **dipakai buat menghapus sesuatu, baik itu file maupun direktori.**

Perintah ini meminta path sebagai argumen, dan kalau path tersebut berupa direktori, maka kita wajib menyertakan flag **-r** yang berarti hapus secara rekursif.

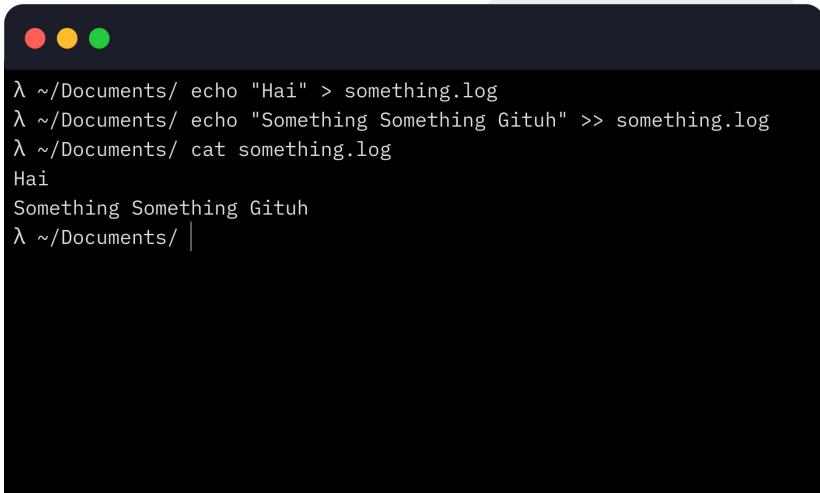
A screenshot of a macOS terminal window. The window has a dark theme with red, yellow, and green close buttons at the top. The terminal itself is black with white text. It shows five commands being run in sequence:

```
λ ~/Documents/ ls cheatsheet  
hidup_bahagia.txt hidup_mewah.txt  
λ ~/Documents/ rm cheatsheet/hidup_mewah.txt  
λ ~/Documents/ rm -r cheatsheet  
λ ~/Documents/
```

## 6. echo

Perintah ini **digunakan untuk mengeluarkan sebuah output di terminal.**

Nggak cuma buat logging, perintah ini bisa memodifikasi output supaya bisa disimpan pada file lain.

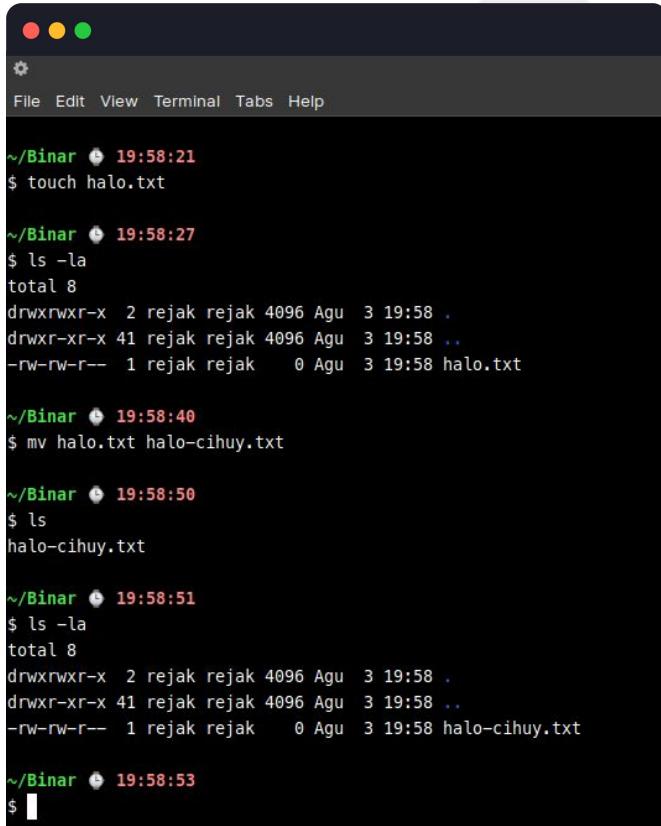
A screenshot of a macOS-style terminal window. The window has three colored circular buttons (red, yellow, green) at the top left. The terminal itself is black with white text. It shows the following command-line session:

```
λ ~/Documents/ echo "Hai" > something.log
λ ~/Documents/ echo "Something Something Gituh" >> something.log
λ ~/Documents/ cat something.log
Hai
Something Something Gituh
λ ~/Documents/ |
```

The cursor is visible at the bottom of the terminal window.

## 7. mv path/ new/path

Perintah ini **digunakan untuk memindahkan satu file dari directory satu ke directory lain atau me rename nama file.**



```
~/Binar ✘ 19:58:21
$ touch halo.txt

~/Binar ✘ 19:58:27
$ ls -la
total 8
drwxrwxr-x  2 rejak rejak 4096 Agu  3 19:58 .
drwxr-xr-x 41 rejak rejak 4096 Agu  3 19:58 ..
-rw-rw-r--  1 rejak rejak     0 Agu  3 19:58 halo.txt

~/Binar ✘ 19:58:40
$ mv halo.txt halo-cihuy.txt

~/Binar ✘ 19:58:50
$ ls
halo-cihuy.txt

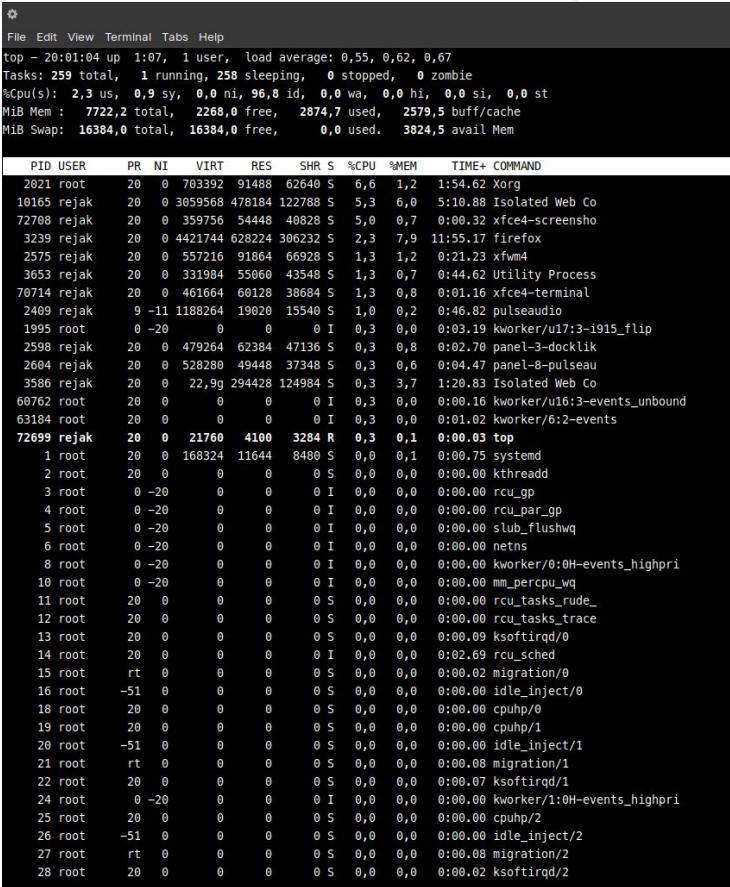
~/Binar ✘ 19:58:51
$ ls -la
total 8
drwxrwxr-x  2 rejak rejak 4096 Agu  3 19:58 .
drwxr-xr-x 41 rejak rejak 4096 Agu  3 19:58 ..
-rw-rw-r--  1 rejak rejak     0 Agu  3 19:58 halo-cihuy.txt

~/Binar ✘ 19:58:53
$
```

## 8. top

Perintah ini **digunakan untuk mengeluarkan informasi task manager lewat terminal.**

Keren kan kaya hacker gitu, gengs 😊

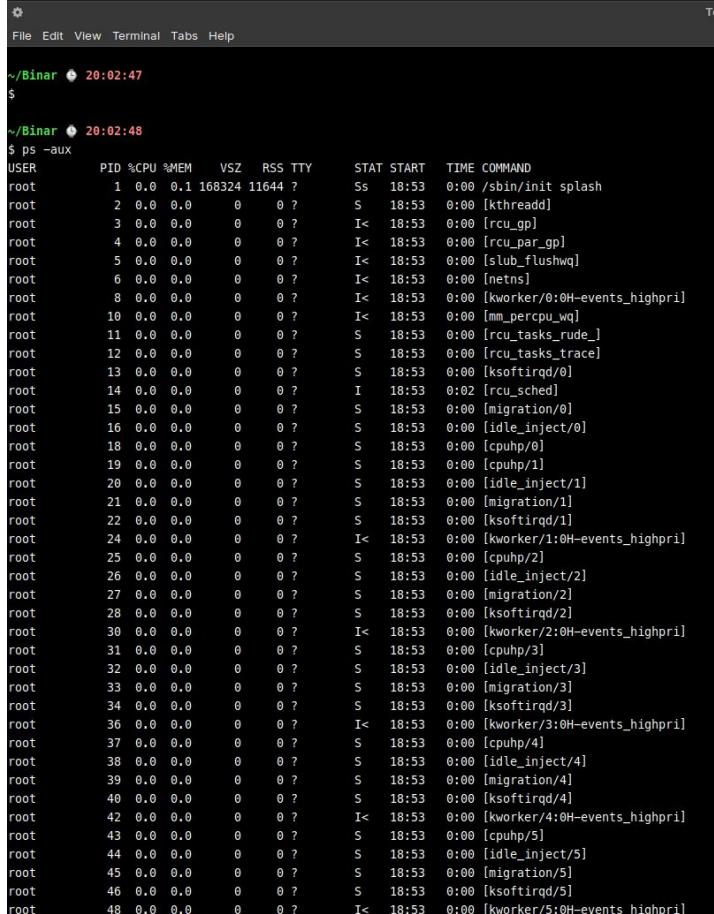


```
File Edit View Terminal Tabs Help
top - 20:01:04 up 1:07, 1 user, load average: 0,55, 0,62, 0,67
Tasks: 259 total, 1 running, 258 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2,3 us, 0,9 sy, 0,0 ni, 96,8 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 7722,2 total, 2268,0 free, 2874,7 used, 2579,5 buff/cache
MiB Swap: 16384,0 total, 16384,0 free, 0,0 used. 3824,5 avail Mem

 PID USER      PR  NI    VIRT   RES   SHR %CPU %MEM TIME+ COMMAND
 2021 root      20   0  703392  91488  62640 S    6,6  1,2 1:54.62 Xorg
10165 rejak    20   0 3059568 478184 122788 S   5,3  6,0 5:10.88 Isolated Web Co
72788 rejak    20   0  359756  54448  40828 S   5,0  0,7 0:00.32 xfce4-screensho
3239 rejak    20   0 4421744 628224 306232 S   2,3  7,9 11:55.17 firefox
2575 rejak    20   0  557216  91864  66928 S   1,3  1,2 0:21.23 xfwm4
3653 rejak    20   0  331984  55060  43548 S   1,3  0,7 0:44.62 Utility Process
70714 rejak    20   0 461664  60128  38684 S   1,3  0,8 0:01.16 xfce4-terminal
2409 rejak    9 -11 1188264 19020  15540 S   1,0  0,2 0:46.82 pulseaudio
1995 root     0 -20   0   0   0 I   0,3  0,0 0:03.19 kworker/u17:3-1915_flip
2598 rejak    20   0 479264  62384  47136 S   0,3  0,8 0:02.70 panel-3-docklik
2664 rejak    20   0  528280 49448  37348 S   0,3  0,6 0:04.47 panel-8-pulseau
3586 rejak    20   0  22,9g 294428 124984 S   0,3  3,7 1:20.83 Isolated Web Co
68762 root     20   0   0   0   0 I   0,3  0,0 0:00.16 kworker/u16:3-events_unbound
63184 root     20   0   0   0   0 I   0,3  0,0 0:01.02 kworker/6:2-events
72699 rejak   20   0  21760  4100  3284 R   0,3  0,1 0:00.03 top
 1 root     20   0 168324 11644  8480 S   0,0  0,1 0:00.75 systemd
 2 root     20   0   0   0   0 S   0,0  0,0 0:00.00 kthreadd
 3 root     0 -20   0   0   0 I   0,0  0,0 0:00.00 rcu_gp
 4 root     0 -20   0   0   0 I   0,0  0,0 0:00.00 rcu_par_gp
 5 root     0 -20   0   0   0 I   0,0  0,0 0:00.00 slub_flushwq
 6 root     0 -20   0   0   0 I   0,0  0,0 0:00.00 netns
 8 root     0 -20   0   0   0 I   0,0  0,0 0:00.00 kworker/0:0H-events_highpri
10 root    0 -20   0   0   0 I   0,0  0,0 0:00.00 mm_percpu_wq
11 root    20   0   0   0   0 S   0,0  0,0 0:00.00 rcu_tasks_rude_
12 root    20   0   0   0   0 S   0,0  0,0 0:00.00 rcu_tasks_trace
13 root    20   0   0   0   0 S   0,0  0,0 0:00.09 ksoftirqd/0
14 root    20   0   0   0   0 I   0,0  0,0 0:02.69 rcu_sched
15 root    rt  0   0   0   0 S   0,0  0,0 0:00.02 migration/0
16 root    -51  0   0   0   0 S   0,0  0,0 0:00.00 idle_inject/0
18 root    20   0   0   0   0 S   0,0  0,0 0:00.00 cpuhp/0
19 root    20   0   0   0   0 S   0,0  0,0 0:00.00 cpuhp/1
20 root    -51  0   0   0   0 S   0,0  0,0 0:00.00 idle_inject/1
21 root    rt  0   0   0   0 S   0,0  0,0 0:00.08 migration/1
22 root    20   0   0   0   0 S   0,0  0,0 0:00.07 ksoftirqd/1
24 root    0 -20   0   0   0 I   0,0  0,0 0:00.00 kworker/1:0H-events_highpri
25 root    20   0   0   0   0 S   0,0  0,0 0:00.00 cpuhp/2
26 root    -51  0   0   0   0 S   0,0  0,0 0:00.00 idle_inject/2
27 root    rt  0   0   0   0 S   0,0  0,0 0:00.08 migration/2
28 root    20   0   0   0   0 S   0,0  0,0 0:00.02 ksoftirqd/2
```

## 9. ps -aux

Perintah ini digunakan untuk **mengeluarkan informasi task apa saja yang sedang berjalan di terminal.**



```
~/Binar ✘ 20:02:47
$ ps -aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START  TIME COMMAND
root      1 0.0  0.1 168324 11644 ?        Ss  18:53 0:00 /sbin/init splash
root      2 0.0  0.0     0   0 ?        S   18:53 0:00 [kthreadd]
root      3 0.0  0.0     0   0 ?        I<  18:53 0:00 [rcu_gp]
root      4 0.0  0.0     0   0 ?        I<  18:53 0:00 [rcu_par_gp]
root      5 0.0  0.0     0   0 ?        I<  18:53 0:00 [slub_flushwq]
root      6 0.0  0.0     0   0 ?        I<  18:53 0:00 [netns]
root      8 0.0  0.0     0   0 ?        I<  18:53 0:00 [kworker/0:0H-events_highpri]
root     10 0.0  0.0     0   0 ?        I<  18:53 0:00 [mm_percpu_wq]
root     11 0.0  0.0     0   0 ?        S   18:53 0:00 [rcu_tasks_rude_]
root     12 0.0  0.0     0   0 ?        S   18:53 0:00 [rcu_tasks_trace]
root     13 0.0  0.0     0   0 ?        S   18:53 0:00 [ksoftirqd/0]
root     14 0.0  0.0     0   0 ?        I   18:53 0:02 [rcu_sched]
root     15 0.0  0.0     0   0 ?        S   18:53 0:00 [migration/0]
root     16 0.0  0.0     0   0 ?        S   18:53 0:00 [idle_inject/0]
root     18 0.0  0.0     0   0 ?        S   18:53 0:00 [cpuhp/0]
root     19 0.0  0.0     0   0 ?        S   18:53 0:00 [cpuhp/1]
root     20 0.0  0.0     0   0 ?        S   18:53 0:00 [idle_inject/1]
root     21 0.0  0.0     0   0 ?        S   18:53 0:00 [migration/1]
root     22 0.0  0.0     0   0 ?        S   18:53 0:00 [ksoftirqd/1]
root     24 0.0  0.0     0   0 ?        I<  18:53 0:00 [kworker/1:0H-events_highpri]
root     25 0.0  0.0     0   0 ?        S   18:53 0:00 [cpuhp/2]
root     26 0.0  0.0     0   0 ?        S   18:53 0:00 [idle_inject/2]
root     27 0.0  0.0     0   0 ?        S   18:53 0:00 [migration/2]
root     28 0.0  0.0     0   0 ?        S   18:53 0:00 [ksoftirqd/2]
root     30 0.0  0.0     0   0 ?        I<  18:53 0:00 [kworker/2:0H-events_highpri]
root     31 0.0  0.0     0   0 ?        S   18:53 0:00 [cpuhp/3]
root     32 0.0  0.0     0   0 ?        S   18:53 0:00 [idle_inject/3]
root     33 0.0  0.0     0   0 ?        S   18:53 0:00 [migration/3]
root     34 0.0  0.0     0   0 ?        S   18:53 0:00 [ksoftirqd/3]
root     36 0.0  0.0     0   0 ?        I<  18:53 0:00 [kworker/3:0H-events_highpri]
root     37 0.0  0.0     0   0 ?        S   18:53 0:00 [cpuhp/4]
root     38 0.0  0.0     0   0 ?        S   18:53 0:00 [idle_inject/4]
root     39 0.0  0.0     0   0 ?        S   18:53 0:00 [migration/4]
root     40 0.0  0.0     0   0 ?        S   18:53 0:00 [ksoftirqd/4]
root     42 0.0  0.0     0   0 ?        I<  18:53 0:00 [kworker/4:0H-events_highpri]
root     43 0.0  0.0     0   0 ?        S   18:53 0:00 [cpuhp/5]
root     44 0.0  0.0     0   0 ?        S   18:53 0:00 [idle_inject/5]
root     45 0.0  0.0     0   0 ?        S   18:53 0:00 [migration/5]
root     46 0.0  0.0     0   0 ?        S   18:53 0:00 [ksoftirqd/5]
root     48 0.0  0.0     0   0 ?        I<  18:53 0:00 [kworker/5:0H-events_highpri]
```

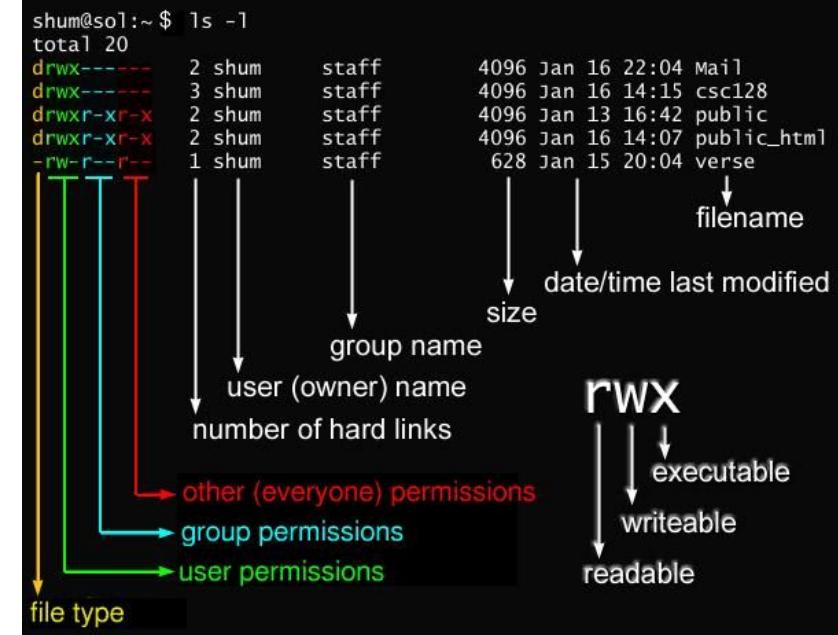
## 10. chmod

Perintah ini **digunakan untuk memodifikasi hak akses dari sebuah folder atau file di terminal.**

Di dalam chmod ada 3 deret rule yaitu user, group dan public. Pada rule, kita dapat mengatur hak aksesnya yaitu:

- **can read**
- **can write**
- **can execute**

Perintah ini akan berguna sekali ketika kita melakukan konfigurasi di dalam server.



The diagram illustrates the output of the command `ls -l` on a Linux system. The output shows a list of files with their permissions, ownership, size, date modified, and names. Arrows point from specific parts of the output to their meanings:

- file type**: Points to the first column of letters (e.g., drwxr-xr-x).
- user permissions**: Points to the first set of three characters in the file type (e.g., rwx).
- group permissions**: Points to the second set of three characters in the file type (e.g., r-x).
- other (everyone) permissions**: Points to the third set of three characters in the file type (e.g., r--).
- user (owner) name**: Points to the second column of names (e.g., shum).
- group name**: Points to the third column of names (e.g., staff).
- number of hard links**: Points to the fourth column of numbers (e.g., 2, 3, 2, 2, 1).
- size**: Points to the fifth column of sizes (e.g., 4096, 4096, 4096, 4096, 628).
- date/time last modified**: Points to the sixth column of dates (e.g., Jan 16, 20:04).
- filename**: Points to the last column of file names (e.g., Mail, csc128, public, public\_html, verse).
- rwx**: Points to the first three characters of the file type (e.g., rwx).
- executable**: Points to the 'x' character in the file type.
- writeable**: Points to the 'w' character in the file type.
- readable**: Points to the 'r' character in the file type.

```
shum@sol:~$ ls -l
total 20
drwxr-xr-x  2 shum  staff   4096 Jan 16 22:04 Mail
drwxr-xr-x  3 shum  staff   4096 Jan 16 14:15 csc128
drwxr-xr-x  2 shum  staff   4096 Jan 13 16:42 public
drwxr-xr-x  2 shum  staff   4096 Jan 16 14:07 public_html
-rw-r--r--  1 shum  staff    628 Jan 15 20:04 verse
```

Nah, tadi itu kegunaan dasar terminal yang wajib diketahui terlebih dahulu.

Nantinya sebagai developer, akan banyak perintah-perintah baru yang akan digunakan untuk menjalankan websitemu masing-masing.

[Bash from Scratch](#) bisa bantu kamu untuk eksplorasi lebih jauh. Silakan dicoba~



### Fun fact tentang bash~

Bash adalah singkatan dari Bourne Again Shell yang merupakan versi baru dari Bourne Shell yang dikembangkan oleh Steve Bourne.

Pemrograman bash shell adalah kumpulan perintah menggunakan script yang ditulis ke dalam bahasa shell, sehingga nantinya dapat dieksekusi oleh sistem operasi.



**Udah paham kan 10 perintah yang paling umum digunakan? Coba Sabrina cek ya, sebutkan 3 perintah dan fungsinya!**



Terminal hanyalah salah satu tools yang digunakan dalam mengembangkan sebuah website. Ada tools lain yang sangat powerful bernama **IDE**.

Yuk langsung kita bahas!

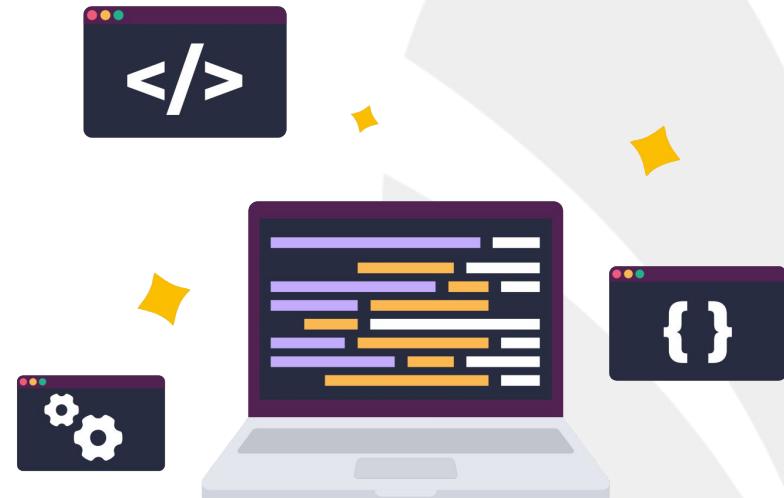


# IDE (Integrated Development Environment)

**Tak perlu risau nggak usah galau,  
ada IDE sebagai solusinya!**

IDE merupakan aplikasi yang menyediakan fasilitas lengkap bagi seorang FSW Developer **yang dikemas dalam satu GUI (Graphic User Interface)**.

Biasanya, sebuah IDE terdiri dari setidaknya dua hal, yaitu **text editor** (untuk melakukan coding) dan **terminal** (untuk menjalankan coding dan operasi file).



# IDE (Integrated Development Environment)

## Terus, fungsinya IDE buat apa?

Nice catch, captain! Okey, biar makin paham, kita pakai contoh kasus berikut 🌟

Sebagai contoh, Sabrina merupakan software developer yang sehari-hari bekerja untuk membuat coding hingga menguji code tersebut.

Dari hulu hingga hilir, Sabrina butuh banyak banget tools seperti text editor, code library, compiler sampai tools untuk menguji software-nya.



# IDE (Integrated Development Environment)

Dalam proses pengembangan software, code harus dipindah dari satu tools ke tools lainnya.

Hal tersebut tentu memakan banyak waktu dan tenaga. Di sinilah **IDE hadir untuk menggabungkan semua fungsi menjadi satu.**

Nggak perlu lagi deh pakai banyak tools untuk berbagai tujuan. Cukup satu saja, semua sudah terintegrasi tanpa repot-repot lagi. Canggih, kan? 😎



## Eits, nggak cuma itu lho 😊

Ada beberapa fitur keren yang bakal bikin kamu makin terpesona sama si IDE ini, antara lain:

1. **Debugger**👉 untuk melakukan tracing atau pencarian terhadap suatu bug.
2. **Source Code Editor**👉 berfungsi untuk menulis code.
3. **Auto Refactoring**👉 Dapat merapikan kode teman - teman dengan otomatis

Dan tentu masih banyak lagi dengan menginstall extensi IDE tertentu!



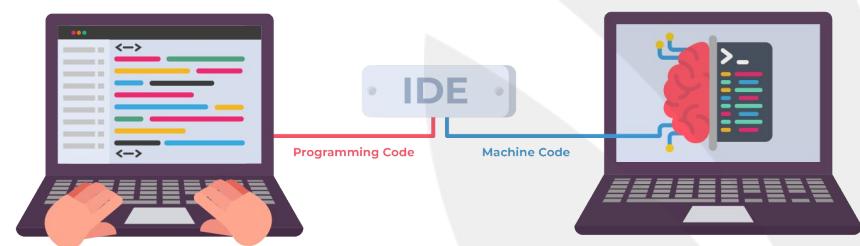
# IDE (Integrated Development Environment)

## Konsep sederhananya kayak gini gengs~

Saat kita menulis sebuah kode, IDE akan mengumpulkan kode tersebut dan menerjemahkannya menjadi MC (Machine Code).

MC ini kode atau bahasa yang hanya bisa terbaca oleh Operating System (OS) seperti IOS, Android, dan lain-lain; yang digunakan oleh website ataupun aplikasi yang dibuat.

Pembahasan lebih dalam tentang MC bisa kamu cek pada link di bawah ini



[What is machine code \(machine language\)? | Definition from TechTarget](#)

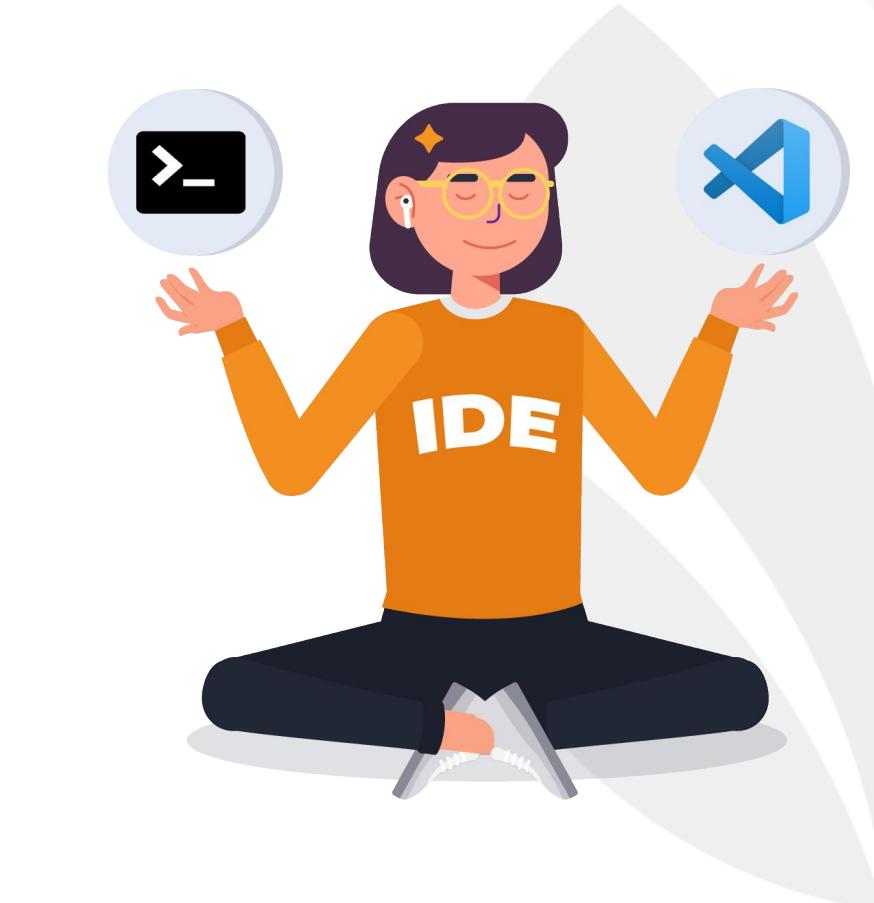
# IDE (Integrated Development Environment)

Nah IDE di dalam software development sendiri biasanya terdiri dari beberapa hal, antara lain:

- **Text editor**
- **Terminal Emulator**

Dua bagian ini ada yang tergabung dalam satu IDE, tapi ada juga yang terpisah.

Kok bisa sih? Yuk kita cari tahu~



# IDE (Integrated Development Environment)

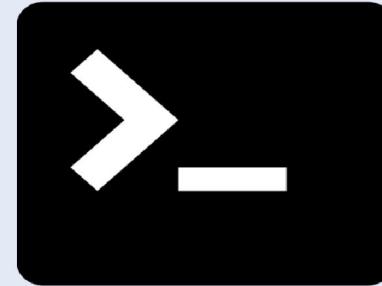
## Terminal Emulator

Kita mulai dulu dengan cari tahu apa itu terminal emulator yaps~

**Terminal emulator itu aplikasi perangkat lunak yang mereplikasi fungsi terminal komputer klasik.**

Biasanya disebut juga sebagai integrated terminal.

Terminal-terminal ini terdiri dari monitor dan keyboard yang digunakan terutama untuk mengakses komputer lain, seperti komputer mini atau mainframe.

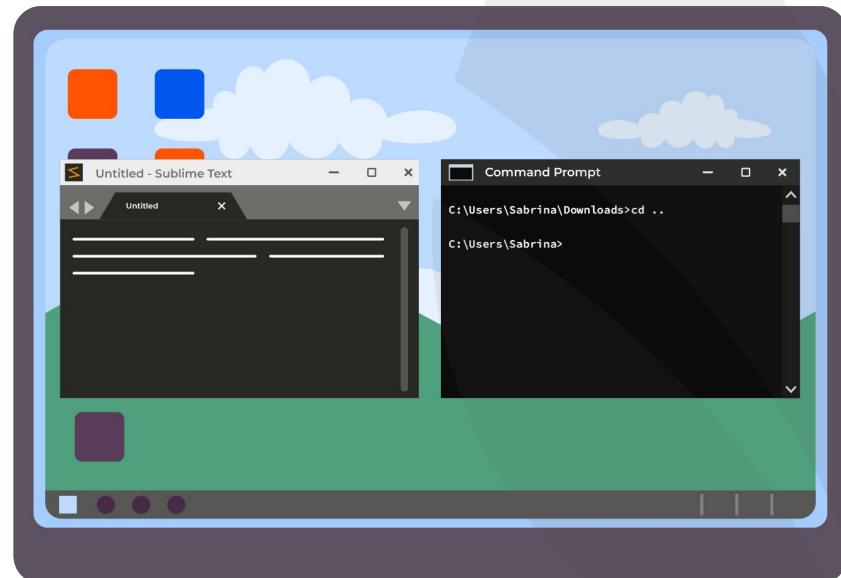


# IDE (Integrated Development Environment)

Uniknya, **nggak semua IDE punya terminal emulator** gengs. Jadi kita perlu buka terminal atau command prompt yang udah tersedia di PC kita.

Contohnya jika kita ingin menggunakan [\*\*Sublime\*\*](#), maka kita perlu buka terminal di luar aplikasi Sublime ini.

Coba perhatikan gambar ya. Di dalam gambar terlihat ada IDE dan juga terminal yang dibuka secara terpisah.



# IDE (Integrated Development Environment)

Nah, **ada juga terminal emulator yang udah tersedia di dalam IDE**. Jadi nggak perlu repot-repot lagi deh buka terminal terpisah~

Contohnya kalau kita menggunakan [Visual Studio Code](#) maka kita nggak perlu lagi buka terminal secara terpisah, karena udah termasuk di dalamnya.

Coba perhatikan gambar ya. Di baris paling kanan adalah contoh terminal yang udah tersedia di IDE.



# IDE (Integrated Development Environment)

## Text Editor

Biasanya digunakan oleh Software Engineer untuk menuliskan code.

Dengan text editor kita bisa memasukkan, mengubah, menyimpan, dan mencetak teks baik berupa angka maupun karakter.

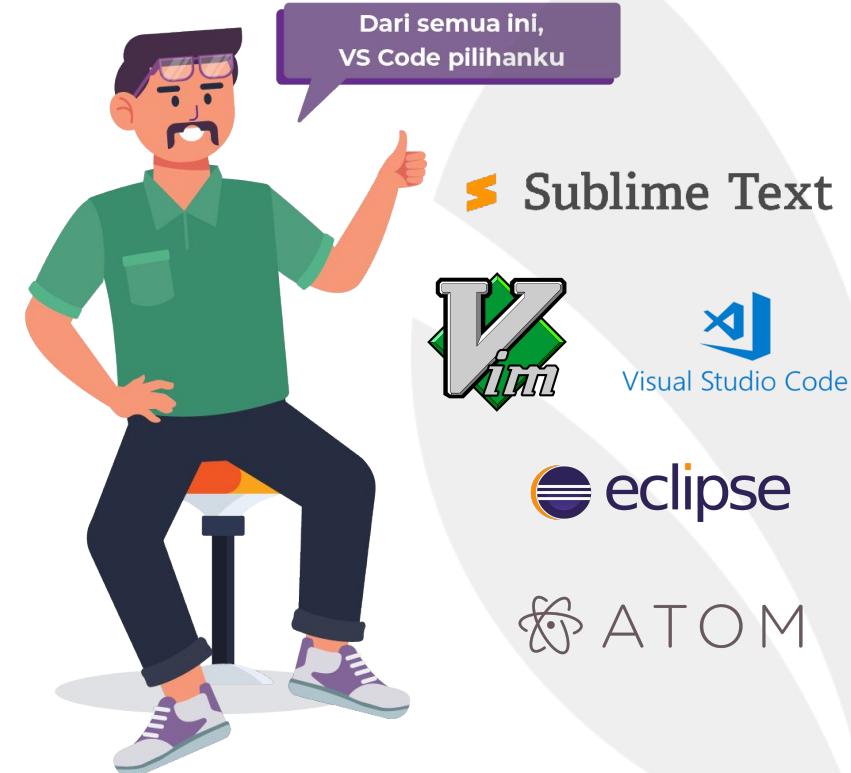


## Ada banyak pilihan text editor~

Berikut beberapa text editor yang populer digunakan developer:

- [Visual Studio Code](#) (untuk bahasa pemrograman php, JavaScript, Python, C++, Flutter, dan lain-lain)
- [Atom](#)
- [Sublime Text](#)
- [Eclipse](#) (untuk bahasa pemrograman Java dan Kotlin)
- [Vim](#)
- dan masih banyak lagi.

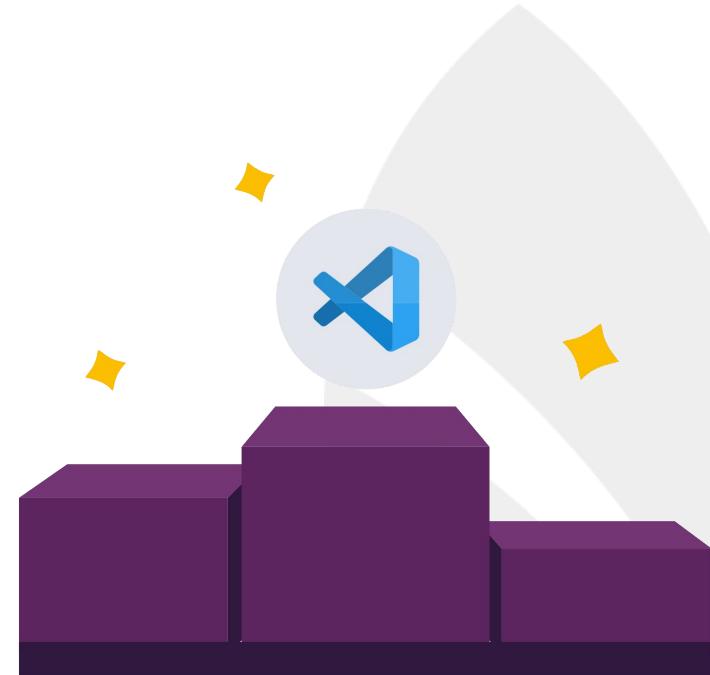
Setiap text editor punya kelebihan dan kekurangan masing-masing. Tapi kalau kamu pemula, kamu bisa mulai coba pakai Visual Studio Code dulu.



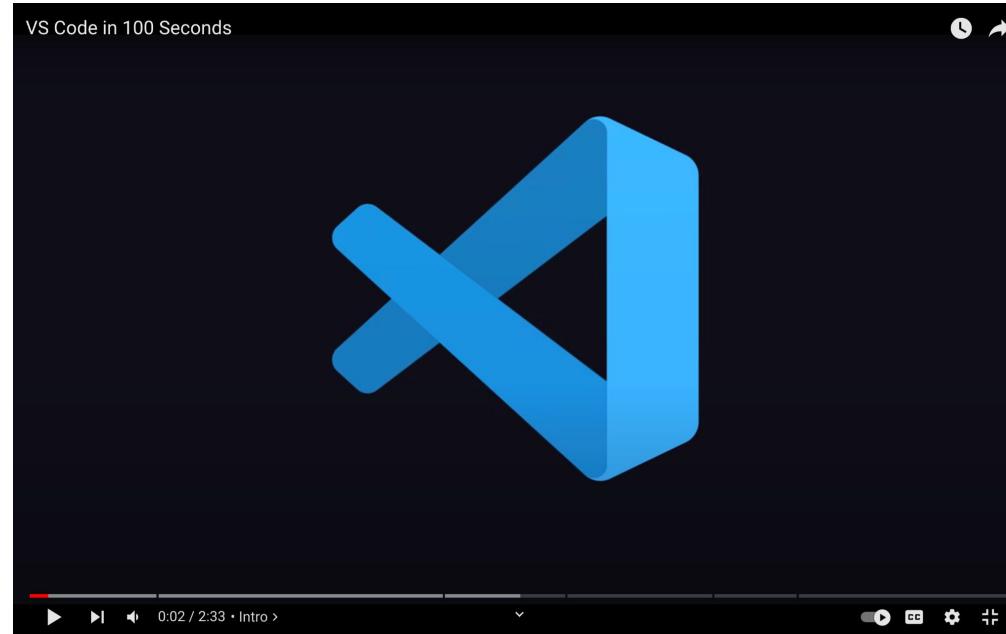
# IDE (Integrated Development Environment)

Berikut **beberapa keunggulan Visual Studio Code** ⤵

- Memiliki integrasi secara langsung dengan terminal (integrated terminal).
- Memiliki banyak opsi [plugin](#) yang bisa dipakai untuk mempermudah proses development saat penulisan kode.
- Memiliki suggestion dan auto-completion, sehingga bisa menghindari kesalahan dalam penulisan kode.
- Memiliki [linter](#), yang berguna untuk mendeteksi letak kesalahan kode.



# IDE (Integrated Development Environment)



Biar kamu lebih paham, kamu bisa menonton video singkat ini buat tahu tentang apa itu VS Code dan apa aja yang bisa kamu lakukan.

[VS Code in 100 Seconds](#)

**Menurut kamu kenapa VS Code jadi salah satu text editor yang paling sering digunakan? Coba comment gengs~**



Kamu pastinya udah download Visual Studio Code kan? Berarti udah siap dong buat mulai ngoding?

Eits tapi, kalau coding mau dilakuin rame-rame bareng tim, teknisnya gimana ya? 😕

Jawabannya, pake Git!



## Apa sih Git itu?

**Git adalah sebuah version control.** Maksudnya, Git adalah sebuah aplikasi yang **berguna untuk mengatur versi dari kode/aplikasi** kita.

Dalam dunia pemrograman, tentu kita akan berkolaborasi dengan programmer lain. Untuk itu, kita perlu sesuatu yang bertanggung jawab guna mengatur tim kita.



Dengan adanya Git, kita bisa dengan mudah berkolaborasi dengan orang lain karena di dalam Git, semua perubahan di dalam kode kita bakal tersimpan ke dalam history.

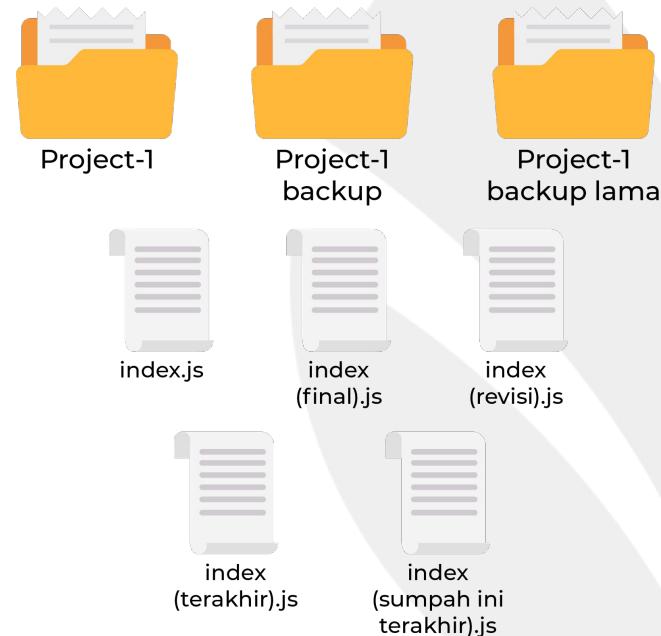
Jadi, pas kita menemui kode, kita bisa melihat history-nya dan siapa yang melakukan perubahan tersebut. Sehingga, kita bisa bisa dengan mudah memperbaikinya. **In a nutshell, Git itu Google Drive-nya Programmer.**



## Bayangkan pas kamu lagi bikin tugas kuliah~

Pas kamu ngerjain tugas kuliah atau skripsi, pasti kamu bakal bikin back-up data, kan? Atau bakal dikerjakan di Google Docs supaya setiap perubahan bisa dilacak.

Jadi, kalau tiba-tiba ada yang kehapus atau salah input, kamu bisa dengan mudah meng-undo-nya!



Nah, programmer juga butuh sesuatu yang bisa nge-backup dan nge-track pekerjaan mereka. Apalagi pekerjaan itu dilakukan bareng-bareng.

Dengan menggunakan Git, pekerjaan akan terlacak dan punya back-up dalam bentuk history.



### Udah kebayang kan fungsi Git? Sekarang, kita lanjut bahas sejarahnya!

Git dibuat oleh Linus Torvalds (pencipta Linux) pas doi membuat proyek linux jadi open source, di mana banyak orang mau berkolaborasi dalam proyek tersebut.



Saking banyaknya orang, sering terjadi perubahan kode dan nggak jelas siapa yang mengubah. Alhasil, permasalahan ini mengganggu pengerjaan proyek Linux.

Makanya, buat mengatasi masalah tersebut, Linux membuat sebuah aplikasi bernama Git yang digunakan untuk mengelola perubahan kode dalam proyek Linux.

**Linus pas bikin  
project open-source  
supaya developer  
bisa berkolaborasi**



**Linus pas tau project  
open-source-nya  
menimbulkan konflik  
antar developer**



**Ternyata Git membantu banget ya dalam penulisan kode. Coba share apa pendapat kamu kalau Git gak dipakai dalam penulisan kode?**



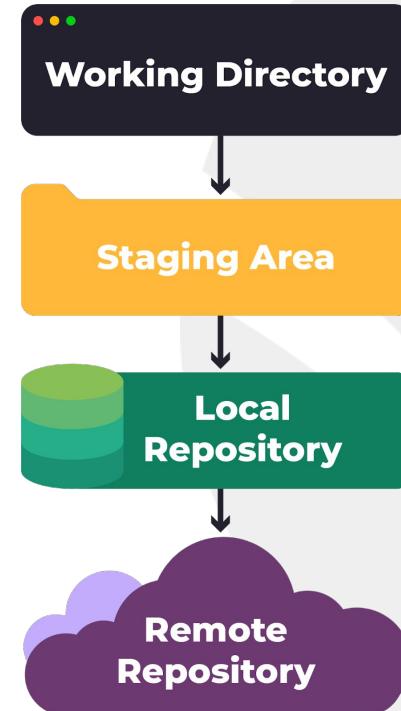
Sebelum kita mempraktekkan penggunaan Git, kita perlu tahu dulu, gimana cara Git menyimpan pekerjaan kita?



Ketika bekerja dengan Git, ia akan menyimpan folder atau pekerjaan kita dalam empat tahap

1. Working Directory
2. Staging Area
3. Local Repository
4. Remote Repository

Kita bahas satu per satu, yuk!



## 1. Working Directory

Sesuai namanya, working directory ini adalah **sebuah direktori buat menyimpan kode yang masih dalam tahap penggerjaan** alias belum selesai

**Misal**, kita lagi mengerjakan sederet baris kode dalam file index.js lalu kita save. Maka, file index.js tersebut bakal berada dalam working directory Git.



## 2. Staging Area

Tahap di mana perubahan yang udah dibuat siap untuk di-commit (dibuatkan berita acara).

**Misal**, kita udah selesai menambahkan sederet baris kode dalam index.js dan ingin melaporkannya kepada developer lain.



# GIT Usage for Project Folder Storage

Buat melaporkannya, kita harus memindahkan index.js dari working directory ke staging area terlebih dahulu.

Intinya, **staging area merupakan tempat buat mengorganisasikan perubahan yang dilakukan untuk dimasukkan dalam berita acara.**



### 3. Local Repository

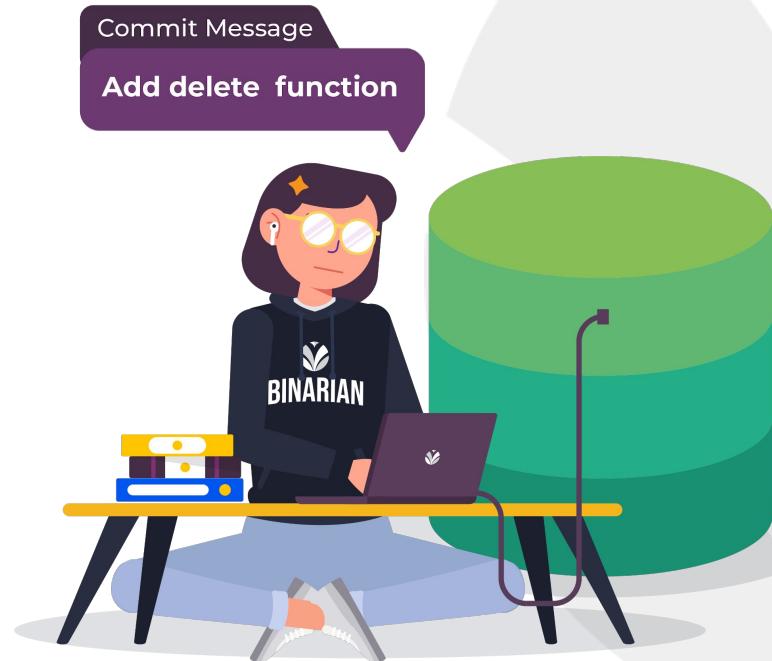
Tahap di mana **perubahan udah ditetapkan (fixed)**, tapi cuma terjadi di dalam komputer (**belum diunggah ke cloud**).

Perubahan yang masuk dalam local repository punya **commit message**, adalah informasi berisi apa yang telah dikerjakan atau diubah dari sebuah file.



# GIT Usage for Project Folder Storage

**Misal**, kamu menambahkan sebuah function delete dalam file index.js. Setelah selesai dikerjakan dan siap disimpan dalam local repository, jangan lupa buat menambahkan keterangan perubahannya atau commit message.



## 4. Remote Repository

Tahap dimana perubahan yang dilakukan udah berada di cloud dan bisa diakses oleh developer lain.

**Misal**, file index.js di local repository bakal kita upload ke remote repository. Nah, perubahan yang terjadi di file index.js tersebut akan otomatis berada di cloud.

Ya kayak pas kita upload tugas ke Google Drive gitu deh~



**Biar lebih paham, mari kita perdalam melalui praktik langsung penggunaan Git!**

Sebelum kita praktik langsung, Sabrina mau ngasih tau dulu nih **beberapa istilah atau perintah-perintah (command) dasar dalam Git** yang masing-masing punya fungsi beda-beda. Ada yang berfungsi membuat branch, menambahkan file, mengupload file, dan lain-lain.

Cus langsung cek di halaman selanjutnya~



# GIT Usage for Project Folder Storage

## BRANCHES.

<code>git branch</code>	Untuk melihat list semua branch lokal
<code>git branch -a</code>	Untuk melihat list semua branch lokal dan branch pada remote repo
<code>git checkout -b branch_name</code>	Membuat branch lokal baru dan langsung pindah ke branch baru tersebut
<code>git checkout branch_name</code>	Pindah ke sebuah branch
<code>git push origin branch_name</code>	Upload semua perubahan pada lokal branch ke remote branch
<code>git branch -m new_name</code>	Mengubah nama sebuah local branch
<code>git branch -d branch_name</code>	Menghapus local branch
<code>git push origin :branch_name</code>	Menghapus remote branch

## LOGS.

<code>git log --oneline</code>	Melihat riwayat commit secara singkat
<code>git diff</code>	Melihat semua changes
<code>git diff myfile</code>	Melihat changes pada file tertentu

## CLEANUP.

<code>git clean -f</code>	Menghapus semua file yang tidak ke track
<code>git clean -df</code>	Menghapus semua file dan folder yang tidak ke track
<code>git checkout -- .</code>	Undo semua changes pada local

Selain perintah di atas, berikut ada [cheatsheet](#) yang bisa membantu kamu dalam bermain git :D

## 1. Instal Git terlebih dahulu

### Untuk Linux

Tahap pertama yang harus kamu lakukan adalah menginstal Git di Linux. Karena Git ini adalah aplikasi di terminal, jadi, untuk instalasinya bisa dilakukan di terminal. **Buka terminal di Linux, lalu ketik perintah di samping ini!**

Untuk MacOS, bakal dijelasin di halaman selanjutnya ya!



```
# Ubuntu/debian based Distro  
sudo apt update  
sudo apt install git -y
```

## Untuk MacOS

**Buka terminal, lalu ketik seperti di bawah ini untuk menginstal brew dan Git.**

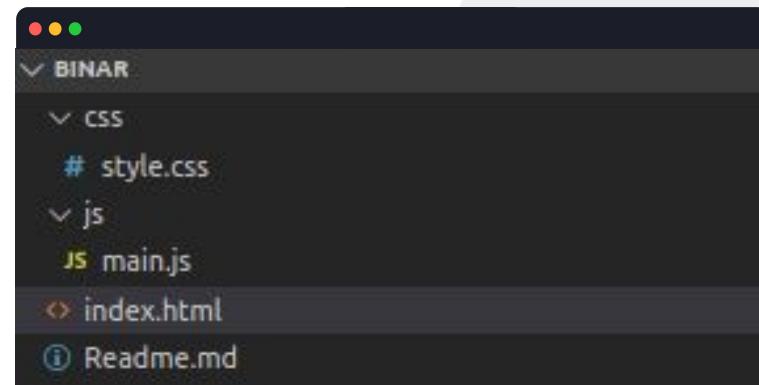


```
# Install brew dulu, yah
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
brew doctor

# Install Git nya
brew install git
```

## 2. Inisialisasi Git di folder project kamu

Misal kamu sedang menjalankan project bernama **BINAR**. Nah, semua file project itu kamu simpan dalam folder bernama **BINAR** seperti gambar di samping ini~



Terus, pas kamu mau mengimplementasikan Git dalam project tersebut, yang harus kamu lakukan adalah:

**Masuk ke terminal, buka folder BINAR lewat terminal, lalu inisialisasi Git dengan mengetik perintah seperti di bawah ini:**



```
cd lokasi/folder/pekerjaan/mu # Untuk pindah lokasi ke folder-mu  
git init # Untuk menginisialisasi git
```

Setelah kamu menginisialisasi Git, maka terminal akan berada dalam folder project kamu.

Selanjutnya, **kamu bisa menjalankan perintah-perintah Git di dalam terminal seperti berikut ini:**

- **git status** - untuk melihat apa saja dokumen update yang ada di repository.
- **git push** - untuk mengirimkan coding yang sudah dikerjakan.
- **git pull** - untuk menarik coding yang ditulis developer lain.
- **git commit** - sebagai penanda sebelum dilakukan git push.



# GIT Usage for Project Folder Storage

Sekarang, coba kamu jalankan perintah `git status`  
Maka, hasilnya akan seperti gambar di samping!

```
● ● ●

On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what
   will be committed)

  README.md
  css/
  index.html
  js/

nothing added to commit but untracked files
present (use "git add" to track)
```

Setelah inisialisasi project, semua file yang berada dalam folder itu akan masuk ke tahap Working Directory. Nah, ada baiknya kamu langsung pindahkan file ke dalam Local Repository dengan melakukan **commit** terlebih dahulu.

Commit pertama ini, disebut **Initial Commit**. Untuk melakukan **commit**, kamu hanya perlu menjalankan dua perintah di samping ini!



```
# Ini akan memindahkan file kita ke Staging Area  
git add .  
# Ini akan memindahkan file kita ke Local Repository  
git commit -m "[Fikri] Initial Commit"
```

## GIT Usage for Project Folder Storage

---

Nah, kalau kamu perhatikan wujud file dan folder kamu di dalam project, keliatannya nggak ada perbedaan sama sekali. Karena, pemindahan stage hanya terjadi di dalam Git itu sendiri, dan kita nggak perlu tau gimana Git melakukan itu.

Tapi intinya, perubahan yang kamu lakukan udah ke catat kok!



## 3. Lakukan perubahan, lalu commit

Setelah selesai melakukan inisialisasi, hal selanjutnya adalah melakukan perubahan di dalam pekerjaan tersebut. Entah menambah kode baru, merubah kode, atau bahkan menghapus kode, pokoknya sesuai dengan kebutuhan pengembangan sistem.

Misal kamu punya file **index.html** dan ingin melakukan perubahan pada file tersebut.



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width, initial-scale=1.0">
    <title>Belajar Git</title>
  </head>
  <body>
    <h1>Hello World</h1>
    <h5>Lagi belajar git nih</h5>
  </body>
</html>
```



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Belajar Git</title>
  </head>
  <body>
    <h1>Hello World</h1>
    <h5>Lagi belajar git nih</h5>
    <h6>Ini perubahan Baru</h6>
  </body>
</html>
```

# GIT Usage for Project Folder Storage

Setelah melakukan perubahan, kamu harus melakukan **commit**. Tujuannya, untuk menandakan bahwa perubahan tersebut telah dicatat oleh Git.

Sebelum melakukan **commit**, pastikan apa yang kamu commit itu **benar**. Kamu bisa jalankan **git status** untuk mengecek apa saja yang berubah. Nanti outputnya akan seperti di pada gambar di samping.

Ini menunjukkan bahwa Git udah melakukan tracking di folder kamu. Silahkan kamu cek ulang apakah file index.html yang kamu ubah sudah benar.



On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will  
be committed)

(use "git checkout -- <file>..." to discard  
changes in working directory)

modified: index.html

no changes added to commit (use "git add"  
and/or "git commit -a")

Kalo udah benar, baru deh sekarang kamu lakukan **commit**. Untuk melakukannya, kamu harus pindahkan perubahan yang kamu lakukan dari **working area** ke **staging area**. Cara memindahkannya mudah, kok. Kamu cukup jalankan perintah ini~

Karena yang kamu ubah adalah file **index.html**, maka kamu harus **pindahkan** file **index.html** ke **staging area**.



```
git add namafileita
```



```
git add index.html
```

# GIT Usage for Project Folder Storage

Nah, selanjutnya kamu bisa cek apakah file index.html tersebut udah dipindah ke staging area. Caranya, **cek menggunakan git status**.

Kalo warnanya udah **hijau**, itu artinya file kamu udah masuk ke **staging area**.



```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   index.html
```

Setelah perubahan masuk ke staging area, selanjutnya kamu harus meng-commit perubahan tersebut. Agar perubahannya dipindah ke **Local Repository**.

```
git commit -m "[Fikri] Perubahan Baru dengan h6"
```

Kamu bisa menggunakan git status untuk memastikan apakah perubahan tersebut sudah di-commit.

```
On branch master
nothing to commit, working tree clean
```

**Nah gampang banget kan? Coba kita  
inget-inget lagi apa aja 3 tahapan  
penggunaan Git? 🤔**



Kita udah bahas penggunaan Git di Local Repository, sekarang kita lanjut gimana penggunaan Git di Remote Repository!

Satu titik dua koma, ayok kita lanjut ke materinya!



### Apa sih bedanya local repository dan remote repository?

Sebelum kita bahas gimana penggunaan Git di remote repository, kamu perlu tahu dulu apa itu remote repository dan apa perbedaannya dengan local repository.

**Perbedaan keduanya terletak di lokasi penyimpanan.**

Kalau **Local repository**, artinya folder tersimpan di device kamu. Kalau **remote repository**, folder tersimpan di cloud dan bisa diakses oleh developer lain.



**Local Repository**



**Remote Repository**

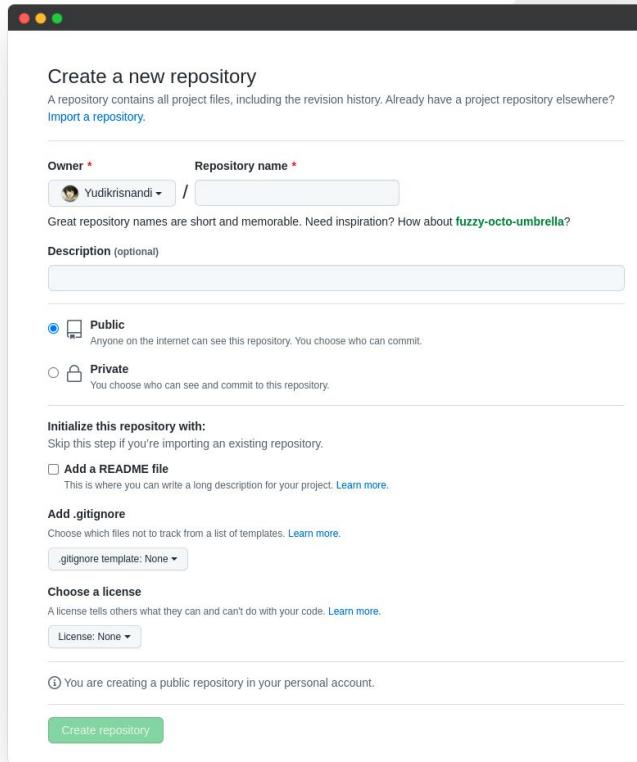
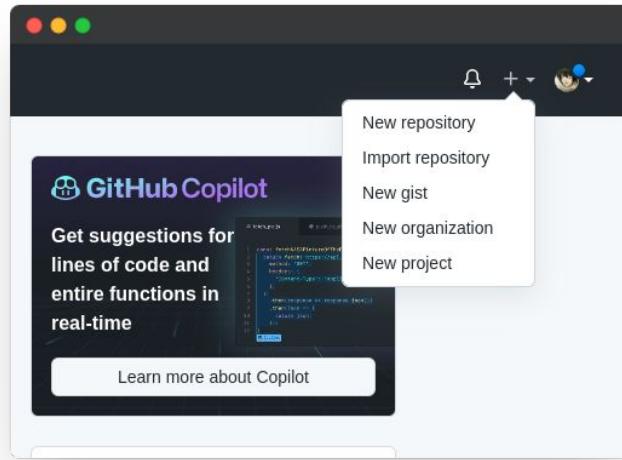
### Berikut ini tiga layanan remote repository yang paling terkenal:

- GitHub
- GitLab
- Bitbucket

Nah, kita bakalan praktek menggunakan **GitHub**. So, pastikan kamu udah membuat akun GitHub, ya! kalau udah punya, kamu bisa langsung membuat repository untuk proyek kamu di GitHub tersebut.

Bingung caranya? Cek slide berikutnya deh~





Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner \* Repository name \*

 Yudikrisnand /

Great repository names are short and memorable. Need inspiration? How about [fuzzy-octo-umbrella](#)?

Description (optional)

 Public Anyone on the internet can see this repository. You choose who can commit.

 Private You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file This is where you can write a long description for your project. [Learn more](#).

Add .gitignore

Choose which files not to track from a list of templates. [Learn more](#).

Choose a license

A license tells others what they can and can't do with your code. [Learn more](#).

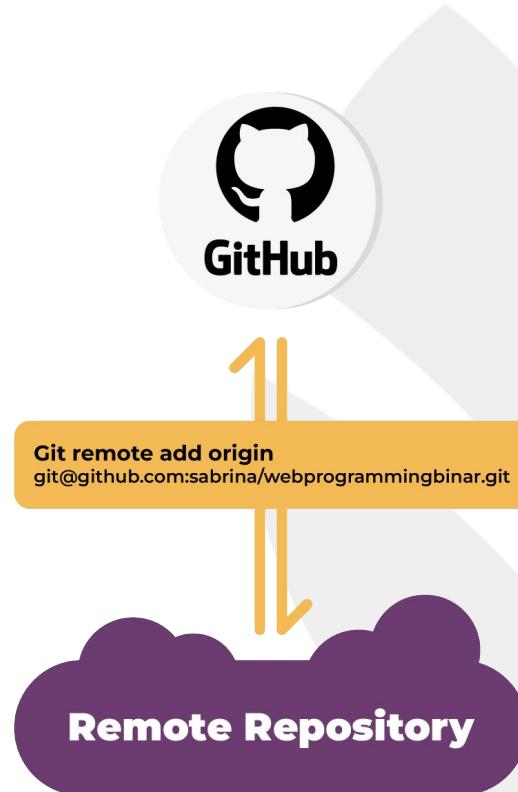
① You are creating a public repository in your personal account.

**Setelah berhasil create project**, kamu perlu menyalin alamat dari repository itu dengan menekan tombol **Clone**. Lalu pilih Clone with HTTPS kalau belum set-up SSH Key.

Untuk menyambungkan ke remote repository, kita bakal pakai perintah Git remote. Perintah ini mengurusi segala sesuatu yang berbau konfigurasi remote repository.

Perintahnya seperti ini: **git remote add origin [git@gitlab.com:namamu/nama-project.git](https://gitlab.com/namamu/nama-project.git)**

Fungsi perintah tadi adalah memberi tahu local repository tentang alamat remote repository yang akan menjadi tujuan mengunggah file.



Selanjutnya, kamu bisa menjalankan suatu perintah untuk mengetahui alamat remote repository, yaitu: **git remote -v**

Untuk menghapus alamat remote repository, kamu bisa menggunakan: **git remote remove origin**

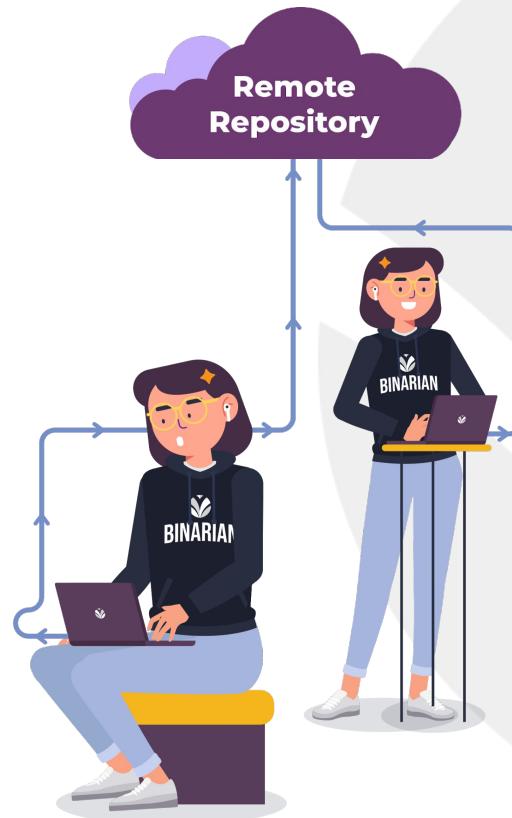
Untuk mengubah alamat remote repository, gunakan perintah ini: **git remote set-url origin git@gitlab.com:alamat/baru.git**



### Git Push dan Git Pull~

Apa tuh? Kayak di pintu minimarket aja~

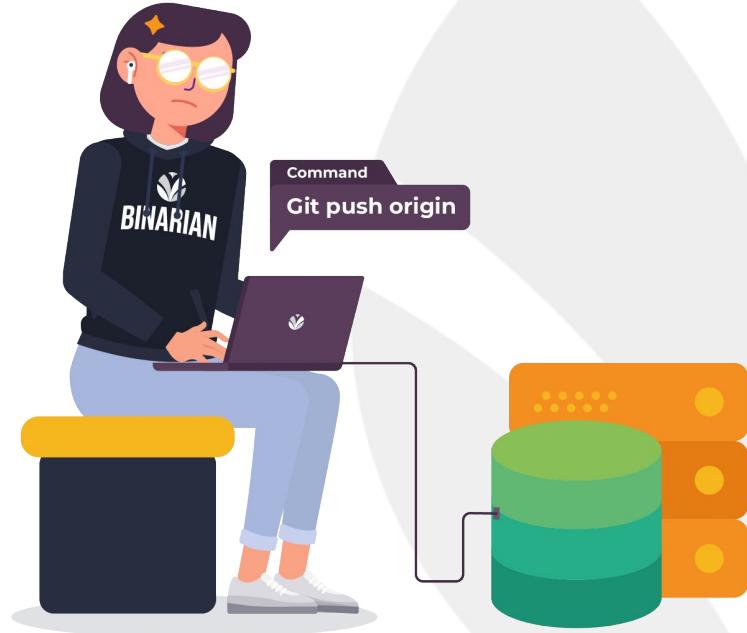
Git Push dan Git Pull ini digunakan agar directory project yang ada di setiap developer merupakan directory yang up-to-date.



**Git Push berfungsi mengunggah pekerjaan ke remote repository.** Caranya, gunakan perintah: **git push origin master**.

Maksud dari perintah ini adalah, kita mengunggah perubahan ke remote repository bernama origin di branch master.

Tapi... sangat nggak dianjurkan untuk mengunggah langsung ke **branch** master, yaa!



### Sekarang, coba kita bahas dulu apa itu Git Pull.

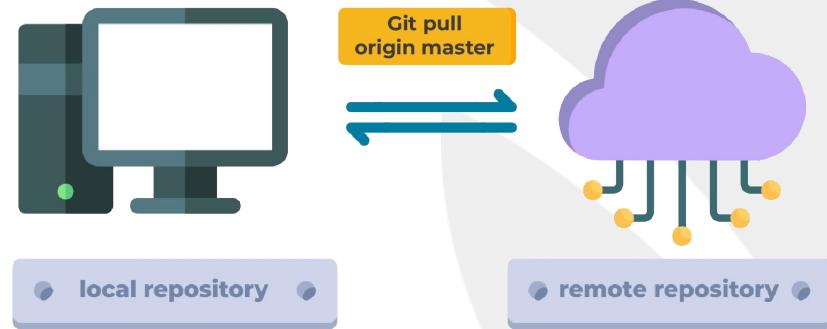
Pas developer lain mengunggah pekerjaan mereka di remote repository, pasti bakal terjadi perubahan kan?

Sebagai sesama tim, udah semestinya perubahan itu relevan dengan pekerjaan kita. Yaa biar sama-sama up-to-date gitu deh!



Di sitalah fungsi Git Pull, yaitu **menyinkronkan perubahan dari remote repository ke local repository.** Caranya, dengan menggunakan perintah: **git pull origin master**

Maksudnya, kita menginstruksikan Git buat mengunduh perubahan terbaru yang terjadi di branch master ke local repository.



**Gimana gengs, akun GitHub-nya udah jadi kan? Kalau udah, boleh dong tipis-tipis cobain bikin repository.**



Sampai sini, mungkin kamu mulai bertanya-tanya, gimana kolaborasi antar developer dilakukan melalui Git? 🤔

Untuk memecahkan masalah tersebut, di dalam Github dibuat yang namanya sistem **branching**, kayak pohon kerja project gitu deh.

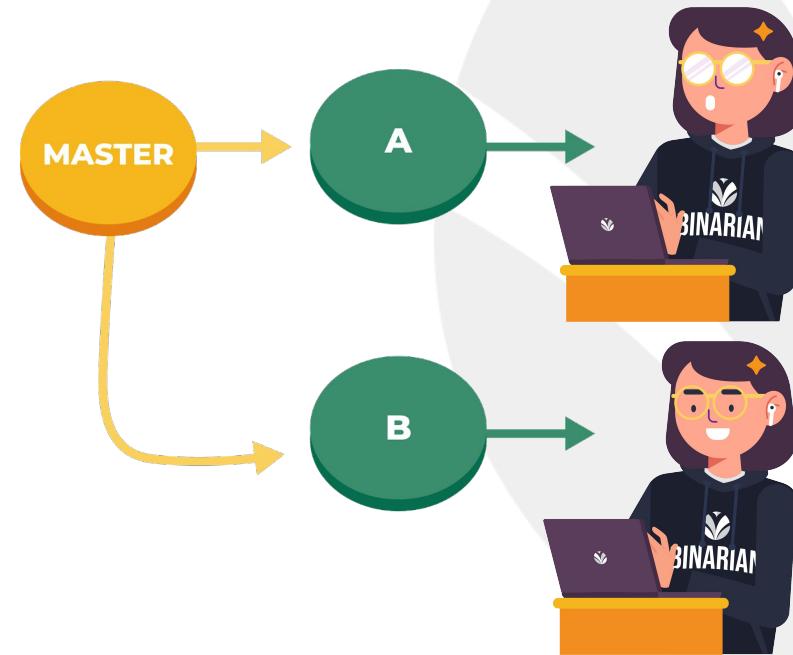
Yuk kita kulik lebih lanjut pembahasannya!



## Berkolaborasi lewat branching~

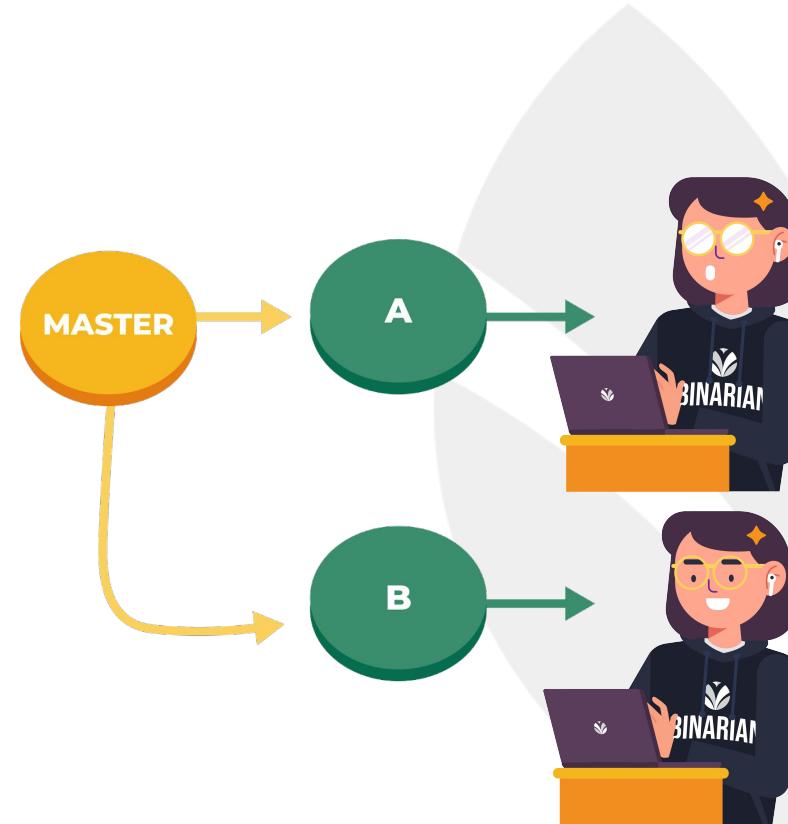
Branch berguna untuk menduplikasi kode di tempat baru, di mana kamu bisa dengan mudah menggabungkan branch A dengan branch B.

Dengan **sistem branching** inilah, kamu **bisa** melakukan kolaborasi dengan developer lain.



Kalau developer A mengerjakan fitur A dan developer B mengerjakan fitur B, mereka berdua bakal mengerjakannya di branch masing-masing. Dengan begitu, nggak akan terjadi tumpang tindih antara kode fitur A dengan fitur B.

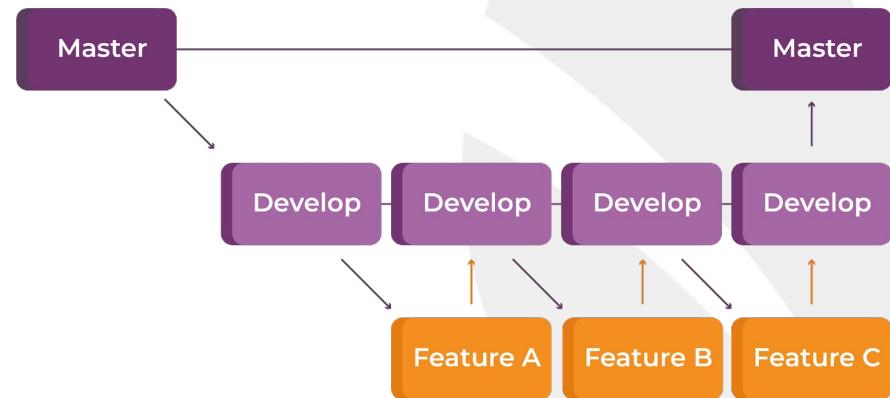
Pas kamu mau membuat suatu fitur atau memperbaiki bug, sangat disarankan buat mengerjakannya di branch lain. Tujuannya supaya nggak terjadi konflik dan lebih mudah buat memonitor perubahan.



Pas kamu membuat project, disarankan juga untuk membuat satu branch bernama "develop".

Soalnya, **branch master cuma dipakai untuk kode yang udah stabil dan udah nggak ada lagi major bugs.**

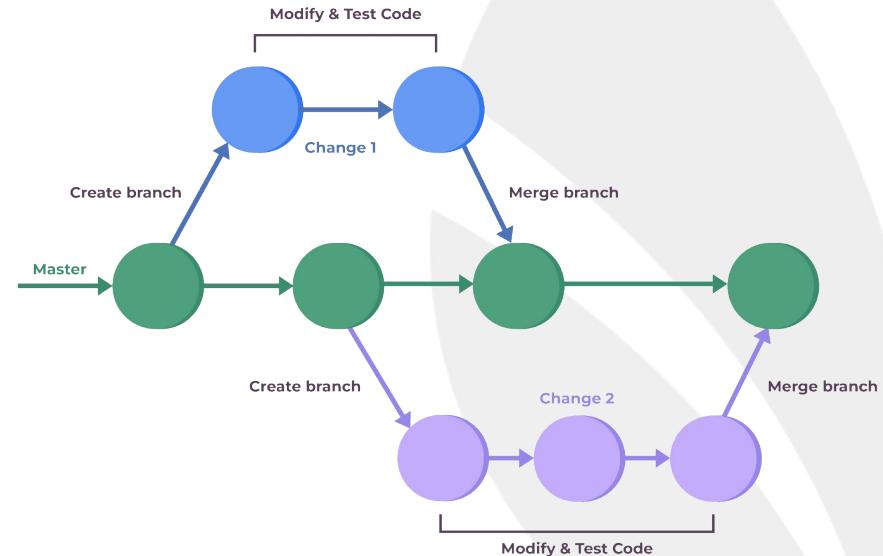
Sedangkan **branch develop berguna untuk menyimpan kode-kode terbaru dan jadi tempat untuk melakukan tes kode**



Cara tadi termasuk umum dilakukan dalam praktik coding, lho. Jadi, sebelum melakukan penggabungan branch develop ke branch master, biasanya akan dilakukan beberapa tes buat memastikan bahwa kode yang dibikin udah stabil.

Kenapa harus kayak gitu?

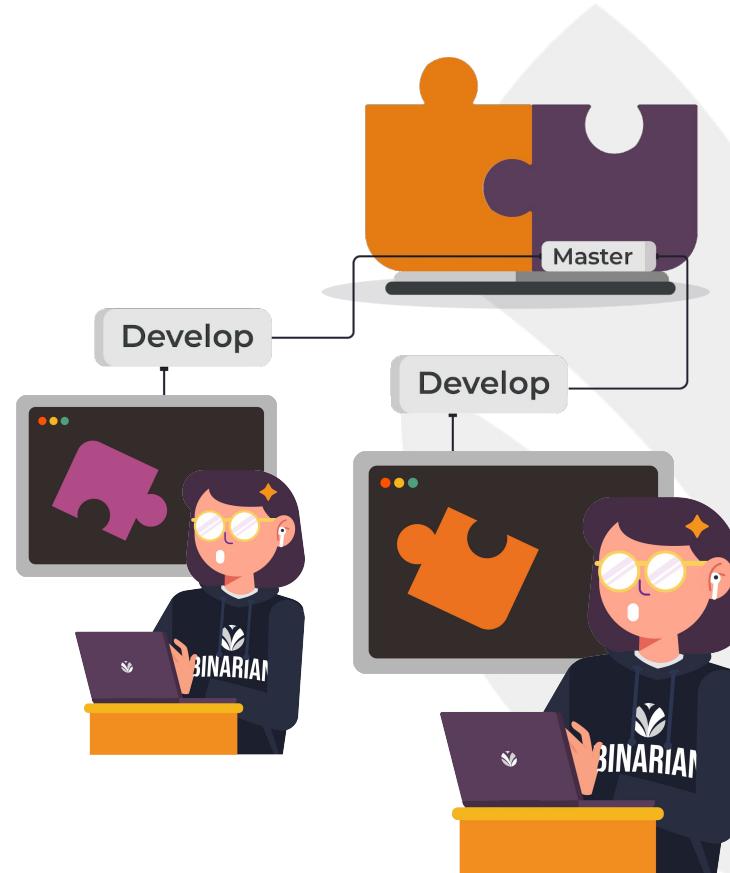
Soalnya, branch master merupakan branch di mana kode akan dipakai langsung oleh user. Bisa gawat kan kalau user menemukan error dalam kode kita~



Pada fase development, biasanya branch develop digunakan sebagai tempat buat menyimpan update fitur terbaru dari kode kamu.

Misal, kamu punya fitur registrasi dan akan mengerjakannya, maka langkah yang harus kamu tempuh adalah:

1. Membuat branch baru bernama **feature/register** dan melakukan perubahan di branch tersebut.
2. Setelah selesai, kamu harus gabungkan branch **feature/register** ke branch **develop**, jangan langsung ke master.
3. Setelah fase development selesai, gabungkan branch **develop** dengan **master**. **Tapi, pastiin kode kamu** udah stabil.



**Kalau udah belajar tentang branching,  
boleh dong coba kolaborasi dengan  
temanmu di kelas..**



Dalam proses pengembangan web, developer juga harus menggabungkan perubahan ke branch develop karena branch release kemungkinan akan berisi berbagai perubahan terbaru.

Eitss, tapi gimana kalau masih ada perubahan kode yang ketinggalan? 🤔

Fenomena ini disebut sebagai **Commit Behind** guys.

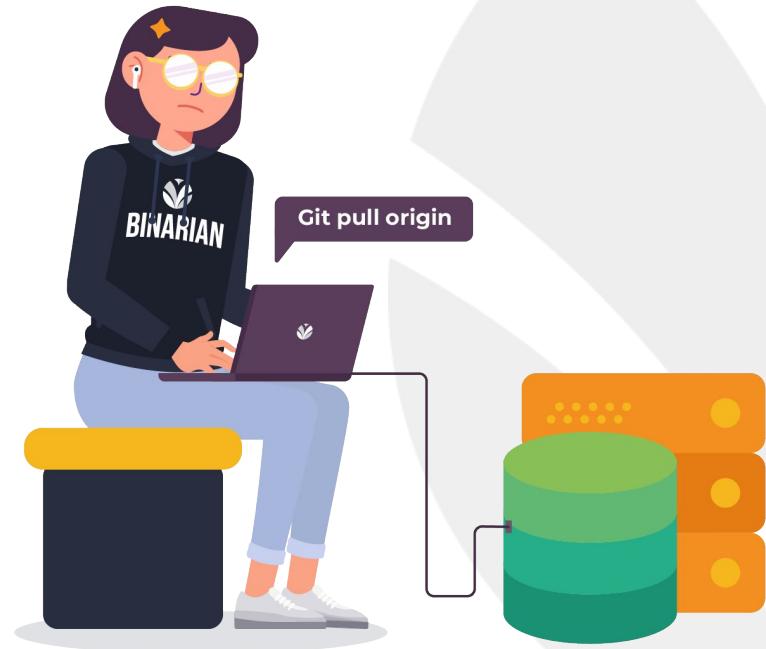


### Commit behind dalam branching~

Pas bekerja dengan branch, mungkin banget terjadi **commit behind**. Apa itu **commit behind**?

Jadi, pas mengajukan penggabungan branch, ada kemungkinan source branch ketinggalan update dari target branch. Untuk itu, source branch harus melakukan **git pull origin target-branch** terlebih dahulu untuk membiasakan update dari target branch.

Itulah yang dimaksud dengan **commit behind**.



### Perlu dicatat nih!

Nggak direkomendasikan banget untuk memaksakan penggabungan ketika ada commit behind.

Kenapa?

Sebabnya, update dari target branch bisa aja terhapus karena source branch belum memilikinya.



### Kayak Sabrina sama Mas Gun ini frens!

Suatu hari, Sabrina sama Mas Gun mengerjakan dua fitur di waktu yang bersamaan. Ternyata, Sabrina kelar duluan.

Setelah itu, Sabrina melakukan penggabungan branch yang ia kerjakan ke branch develop. Branch develop pun mendapat perubahan dari fitur yang dikerjakan Sabrina.



Nggak lama berselang, Mas Gun juga kelar. Abis itu, Mas Gun melakukan penggabungan ke branch develop tanpa konfirmasi ke Sabrina.

Akibatnya, kode jadi berantakan karena fitur yang dikerjakan Mas Gun belum update dengan fitur yang dikerjakan Sabrina.



### Mas Gun harusnya lakuin git pull origin dulu 😞

Sebelum melakukan penggabungan, Mas Gun harusnya ngelakuin **git pull origin target-branch** terlebih dahulu buat ngatasin commit behind. Biar source branch (branch-nya Mas Gun) dapat update dari target branch.



### Nasi udah jadi bubur 🍲

Kalau udah terlanjur digabung, harus gimana dong?  
Dengan kata lain, gimana kalau kita gagal saat merging  
kode karena adanya perbedaan script?

Kondisi gagalnya merging kode ini sering terjadi karena  
perubahan kode di target branch nggak bisa dihindari.  
So, gimana solusinya?

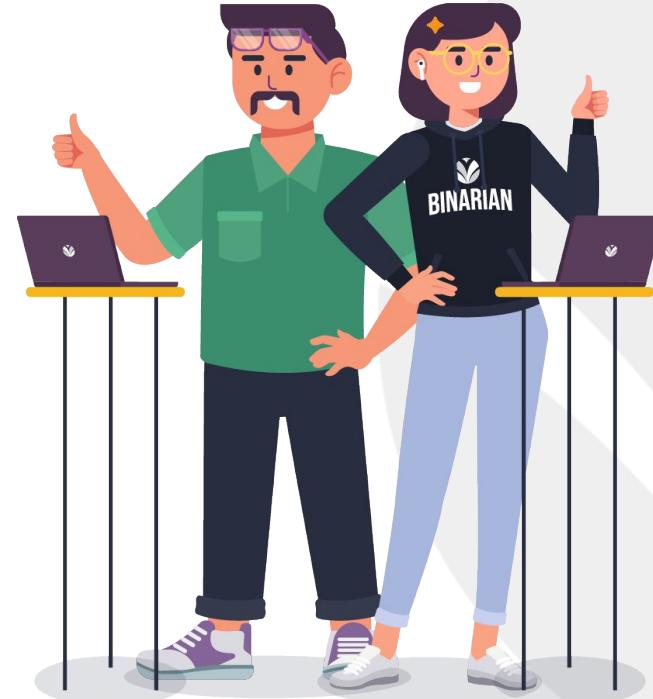


### Stay calm! Solusinya bisa bikin branch khusus 🔥

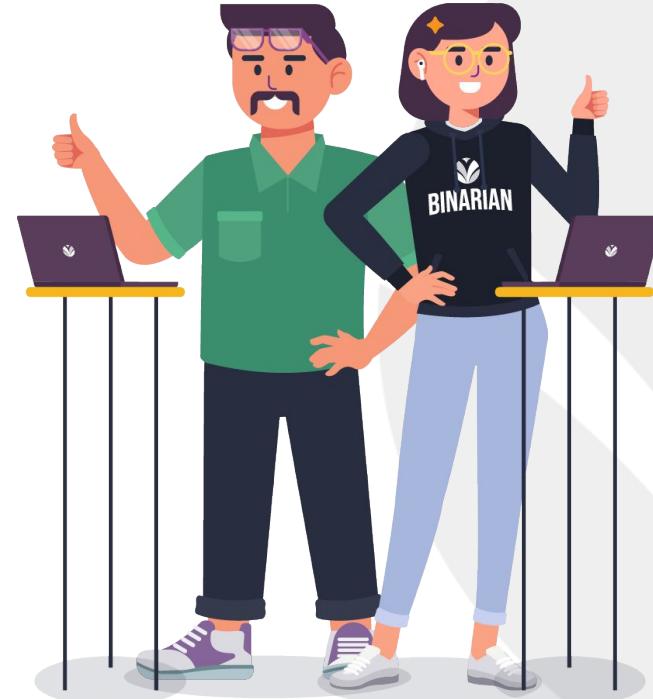
Kita bisa bikin branch khusus buat mengatasi gagalnya merging kode.

Caranya kayak gini frens:

1. **Checkout/pindah ke target branch.** Dalam kasus Mas Gun tadi, ia bisa pindah ke branch develop yang menjadi tujuan merge kodennya.



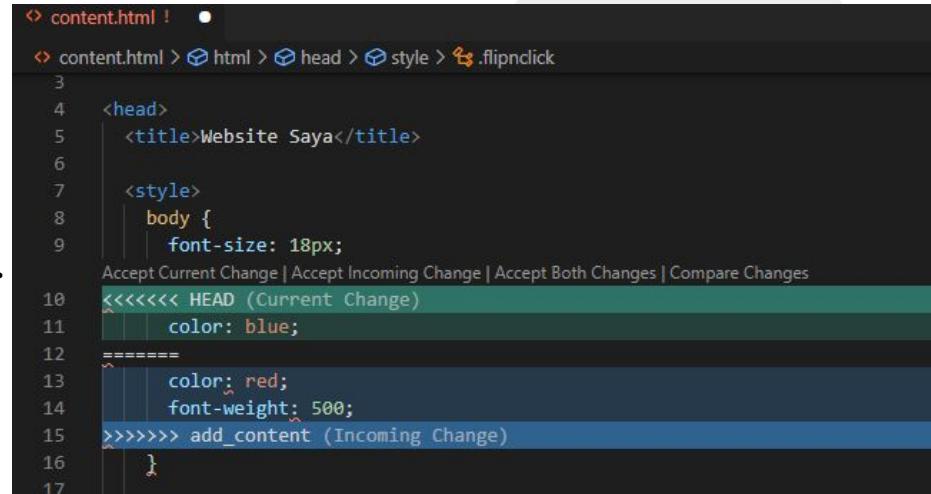
2. **Buatlah branch baru sebagai tempat resolve conflict.** Dalam kasus Mas Gun, ia harus membuat branch baru sebagai tempat resolve conflict dari branch develop yang udah di-update Sabrina.
3. **Lakukan merging branch fitur ke branch resolve yang udah dibikin. Caranya, pilih kode yang mau diambil.** Dalam kasus Mas Gun, ia akan mengambil kode baru yang berisi fitur yang udah dikerjakan Sabrina.



Misalnya kayak gini! Saat melakukan merging, terlihat conflict seperti pada gambar! 👉

Nah, saat membandingkan kode, kamu bisa pilih **current change, incoming change**, atau **both changes**.

FYI, current change adalah **branch kita saat ini**.  
Incoming change adalah **branch tujuan**.



The screenshot shows a code editor with the file `content.html` open. The code is as follows:

```
<head>
  <title>Website Saya</title>
<style>
  body {
    font-size: 18px;
  }
  color: blue;
  color: red;
  font-weight: 500;
}</style>
```

A conflict is visible between line 11 and line 13. Line 11 is highlighted in green and labeled "HEAD (Current Change)". Line 13 is highlighted in red and labeled "add\_content (Incoming Change)". A horizontal line separates the two sections.

4. **Gabungkan branch resolve tadi dengan branch develop.** Caranya, dengan kembali ke terminal dan lakukan commit 

Kalau udah, jadi lengkap deh dalam branch develop. Ada kode fitur asal, fitur Sabrina, dan fitur Mas Gun. 



```
Git add .
Git commit -m 'pesan commit'
Git push
```

**Yosh, selesai perjalanan kita belajar  
Terminal, IDE, dan GIT.**

**Menurut kamu, dari ketiga tools tersebut,  
mana yang paling krusial? Mengapa? 🤔**



Saatnya  
Mini  
Challenge

### Study Case 1

Menurutmu, apakah dalam pengembangan aplikasi  
membutuhkan terminal?  
Coba diskusi bersama kelompokmu ya!

### Mini Challenge 1: Penggunaan Terminal

## Saatnya latihan menggunakan terminal!

Yuk buka terminalmu lalu lakukan perintah-perintah di bawah ini!

1. Buka terminal yang ada di komputermu.
2. Buat folder dengan nama mu. Contoh <nama-mu>.
3. Masuk ke folder <nama-mu> sebelumnya.
4. Lalu buat file dengan nama `log.txt`
5. Isi file `log.txt` dengan content yang ada di sini ya [Log example](#) dengan editor favoritmu (nano, vim etc) lalu simpan.
6. Baca isi content dari file `log.txt` untuk memastikan file berisi content.
7. Buat folder baru dengan nama `binar-cli`.
8. Pindahkan file `log.txt` ke folder `binar-cli`.

### Study Case 2

Menurutmu, ketika menemui kode yang conflict dari setelah pull, apa yang akan kamu lakukan?  
Coba diskusi bersama kelompokmu ya!

### Study Case 3

Menurutmu, apakah code review dalam proses pull request/merge request itu perlu?  
Coba diskusi bersama kelompokmu ya!

Setelah mempelajari topic ini, kamu bisa mulai mengerjakan challenge di bagian “**Membuat sebuah folder khusus bernama challenge-01**”. Jika masih ada pertanyaan, jangan ragu untuk menghubungi facilitator ya!

Aku pasti bisa



## Rangkuman

### Mengenal konsep dan fungsi Terminal

Terminal berfungsi untuk mengembangkan website dengan memastikan apakah kode kita berjalan dengan baik serta memastikan tampilan website kita sesuai dengan desainnya. Dengan terminal kita bisa menulis perintah untuk menjalankan komputer tanpa perlu GUI.

### Mengenal konsep dan fungsi IDE

IDE (Integrated Development Environment) Merupakan aplikasi yang digunakan menulis kode untuk website yang mau kita buat.

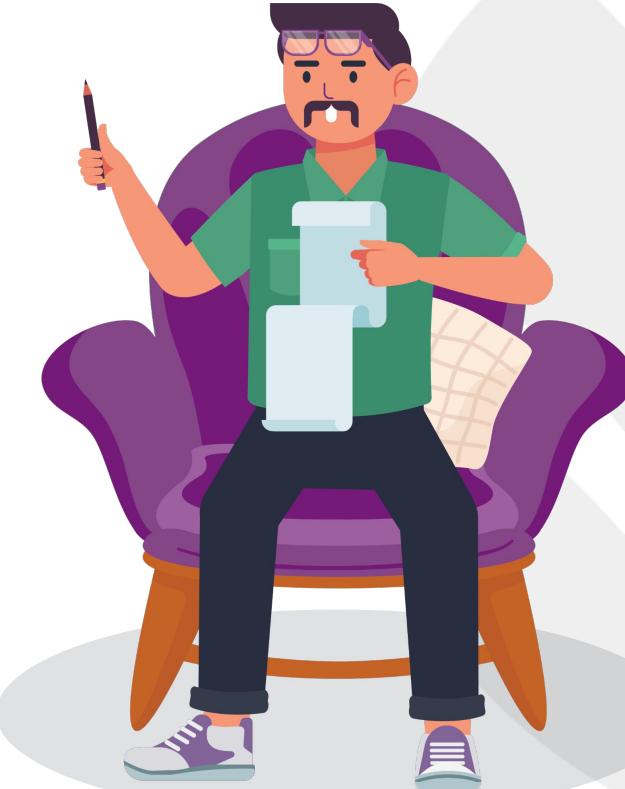


### Definisi Git

Git adalah sebuah version control. Maksudnya, Git adalah sebuah aplikasi yang berguna untuk mengatur versi dari kode/aplikasi kita.

### Tahap penyimpanan Git

Working Directory → Staging Area → Local Repository → Remote Repository



## Perintah dasar dalam Git

### BRANCHES.

<code>git branch</code>	Untuk melihat list semua branch lokal
<code>git branch -a</code>	Untuk melihat list semua branch lokal dan branch pada remote repo
<code>git checkout -b branch_name</code>	Membuat branch lokal baru dan langsung pindah ke branch baru tersebut
<code>git checkout branch_name</code>	Pindah ke sebuah branch
<code>git push origin branch_name</code>	Upload semua perubahan pada lokal branch ke remote branch
<code>git branch -m new_name</code>	Mengubah nama sebuah local branch
<code>git branch -d branch_name</code>	Menghapus local branch
<code>git push origin :branch_name</code>	Menghapus remote branch

### LOGS.

<code>git log --oneline</code>	Melihat riwayat commit secara singkat
<code>git diff</code>	Melihat semua changes
<code>git diff myfile</code>	Melihat changes pada file tertentu

### CLEANUP.

<code>git clean -f</code>	Menghapus semua file yang tidak ke track
<code>git clean -df</code>	Menghapus semua file dan folder yang tidak ke track
<code>git checkout -- .</code>	Undo semua changes pada local

### Praktek dalam Git

- **git status** - untuk melihat apa saja dokumen update yang ada di repository.
- **git push** - untuk mengirimkan coding yang sudah dikerjakan.
- **git pull** - untuk menarik coding yang ditulis developer lain.
- **git commit** - sebagai penanda sebelum dilakukan git push.



## Referensi

1. [Bash From Scratch](#)
2. [Beginner's Guide To The Linux Terminal](#)



Yuhuuu! Kamu udah berhasil  
menamatkan **Chapter 1 Topic 2** 🎉

Eitsss, tapi materinya masih berlanjut nih.

Sampai jumpa di **Topic 3** ya, learners!

