

git - panduan ringkas



want a simple
but powerful
git client for
your mac?

persiapan

[Unduh git untuk OSX](#)

[Unduh git untuk Windows](#)

[Unduh git untuk Linux](#)

buat repositori baru

buat lah direktori baru, buka dan jalankan

`git init`

untuk membuat repositori git baru.

periksa repositori

buat lah salinan kerja dari repositori lokal dengan menjalankan perintah

`git clone /jalur/ke/repositori`

saat menggunakan server jarak-jauh, perintahnya menjadi

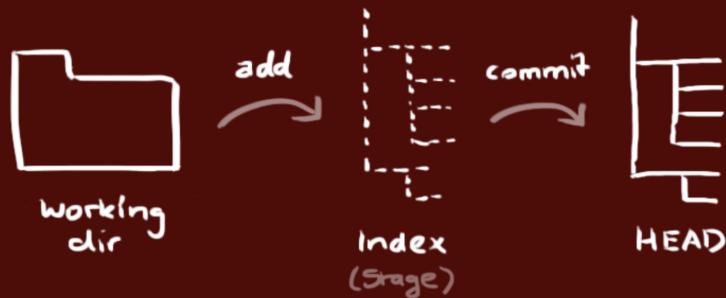
~~`git clone nomorengangus@host:/jalur/ke/repositori`~~

```
git clone namaperiggunaanost:/jatur/repositori
```

alur-kerja

repositori lokal kamu terdiri dari tiga bagian pokok yang disebut "trees"

dikelola oleh git. yang pertama adalah **Direktori Kerja** yang menyimpan berkas aktual. kedua adalah **Indeks** yang berperan sebagai pengolah data dan terakhir **HEAD** yang mengarah pada komit terakhir.



tambah & komit

kamu bisa melakukan perubahan (penambahan ke **Indeks**)

menggunakan

```
git add <namaberkas>
```

```
git add *
```

Ini merupakan langkah awal alur-kerja dasar git. Untuk komit

sepenuhnya gunakan

```
git commit -m "Pesanan komit"
```

Sekarang berkas telah berkomit di **HEAD**, tapi belum di repositori jarak-jauh.

mengirim perubahan

Saat ini perubahan telah tersimpan di **HEAD** salinan kerja lokal kamu.

Untuk mengirimkannya ke repositori jarak-jauh, lakukan

```
git push origin master
```

Ubah *master* sesuai cabang yang kamu inginkan.

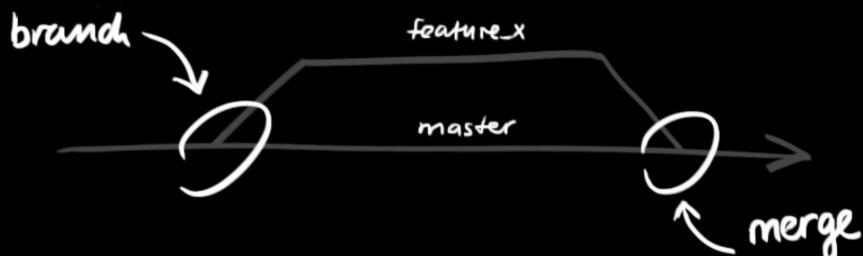
Jika repositori yang ada belum dikloning dan ingin dihubungkan ke server jarak-jauh, kamu perlu menambahkan

```
git remote add origin <server>
```

Sekarang kamu bisa mengirimkan perubahan ke server jarak-jauh yang dituju

percabangan

percabangan atau *branching* digunakan untuk mengembangkan fitur-fitur secara terisolasi. Cabang utama atau *master* merupakan cabang bawaan ketika kamu membuat repositori. Gunakan cabang lain untuk pengembangan, setelah selesai, gabungkan kembali ke cabang utama.



buat cabang baru dengan nama "fitur_x" dan beralih kedalamnya

menggunakan

```
git checkout -b fitur_x
```

beralih kembali ke *master*

```
git checkout master
```

dan hapus cabang yang tadi dibuat

```
git branch -d fitur_x
```

suatu cabang *tidak terbuka untuk yang lainnya* kecuali jika kamu

mengirimkannya ke repositori jarak-jauh.

```
git push origin <cabang>
```

perbaru & gabung

untuk memperbarui repositori lokal ke komit terkini, lakukan

```
git pull
```

dari direktori kerja kamu untuk *mengambil* dan *menggabungkan* perubahan jarak-jauh.

untuk menggabungkan cabang lain ke cabang aktif (misal *master*),

gunakan

```
git merge <cabang>
```

pada kasus diatas, git mencoba menggabungkan perubahan secara otomatis. Sayangnya hal ini tak selalu berjalan mulus dan bisa

menyebabkan *konflik*. Kamu lah yang bertanggung jawab menggabungkan *konflik* tersebut secara manual dengan menyunting berkas yang ditunjukkan git. Setelah itu, kamu perlu memmarkahinya

dengan

```
git add <namaberkas>
```

sebelum penggabungan berlaku, kamu bisa melakukan pratinjau menggunakan

```
git diff <cabang_asal> <cabang_tujuan>
```

menandai

sangat dianjurkan membuat penanda atau *tags* untuk perangkat lunak yang dirilis. Hal ini amat lah lazim, yang juga terjadi di SVN. Kamu bisa membuat penanda baru dengan nama *1.0.0* dengan menjalankan

```
git tag 1.0.0 1b2e1d63ff
```

1b2e1d63ff adalah 10 karakter pertama dari identitas komit yang ingin kamu referensikan ke penanda. Kamu bisa mendapatkan identitas komit dengan melihat...

log

dalam bentuknya yang paling sederhana, kamu bisa mempelajari

riwayat repositori menggunakan.. `git log`

kamu bisa menambahkan banyak parameter untuk menampilkan log sesuai keinginan. Untuk melihat komit penulis tertentu:

```
git log --author=bob
```

Untuk melihat log yang dimampatkan, satu baris per komit:

```
git log --pretty=oneline
```

Atau mungkin kamu ingin melihat pohon *ASCII art* seluruh percabangan disertai nama dan penandanya:

```
git log --graph --oneline --decorate --all
```

Sekedar melihat berkas yang berubah:

```
git log --name-status
```

Ini baru sedikit saja dari sekian banyak parameter yang bisa kamu gunakan. Lebih jauh lagi, lihat `git log --help`

mengembalikan perubahan lokal

Seandainya kamu melakukan kesalahan (yang tentunya tak pernah terjadi ;) kamu bisa mengembalikannya menggunakan perintah

```
git checkout -- <namaberkas>
```

perintah di atas mengembalikan perubahan di dalam pokok kerja kamu dengan konten terakhir dari *HEAD*. Perubahan dan berkas baru yang telah ditambahkan ke indeks akan tetap tersimpan.

Jika kamu ingin menggugurkan perubahan dan komit lokal seutuhnya, ambil riwayat terakhir dari server dan arahkan ke cabang *master* lokal seperti ini

```
git fetch origin
```

```
git reset --hard origin/master
```

petunjuk berguna

GUI git bawaan

```
gitk
```

menggunakan output git penuh warna

```
git config color.ui true
```

menunjukkan log satu baris per komit

```
git config format.pretty oneline
```

menggunakan penambahan interaktif

```
git add -i
```

