MODUL 7

GIT

A. TUJUAN PRAKTIKUM

a. Mahasiswa mengenal teknik versioning dengan Git
b. Mahasiswa mampu berkolaborasi dalam tim dengan Git
c. Mahasiswa mampu mengembangkan aplikasi perangkat lunak dengan Git

B. ALOKASI WAKTU 1 x 50 menit

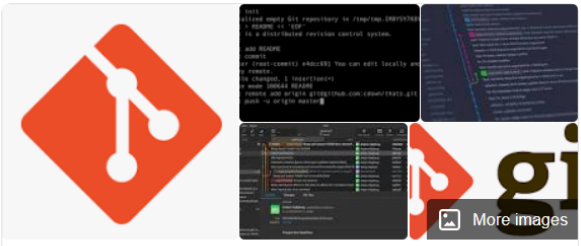
C. DASAR TEORI

1. Git

Git adalah *version control system* yang digunakan para developer untuk mengembangkan software secara bersama-bersama. Fungsi utama git yaitu mengatur versi dari source code program anda dengan memberi tanda baris dan code mana yang ditambah diganti atau dihapus.

Git ini sebenarnya memudahkan programmer untuk mengetahui perubahan source codenya daripada harus membuat file baru seperti Program.java, index.html, index2.html, index3.html. Selain itu, dengan git kita tak perlu khawatir code yang kita kerjakan bentrok, karena setiap developer dapat membuat branch sebagai workspacenya.

Fitur yang tak kalah keren lagi di dalam Git adalah kita dapat memberi komentar pada source code yang telah ditambah/diubah, hal ini mempermudah developer lain untuk tahu kendala apa yang dialami developer lain.



Git

System software

Git is a distributed version-control system for tracking changes in any set of files, originally designed for coordinating work among programmers cooperating on source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows. Wikipedia

Developer(s): Junio Hamano and others

Stable release: 2.29.2 / 29 October 2020; 5 days ago

Original author: Linus Torvalds

Operating system: POSIX (Linux, macOS, Solaris, AIX), Windows

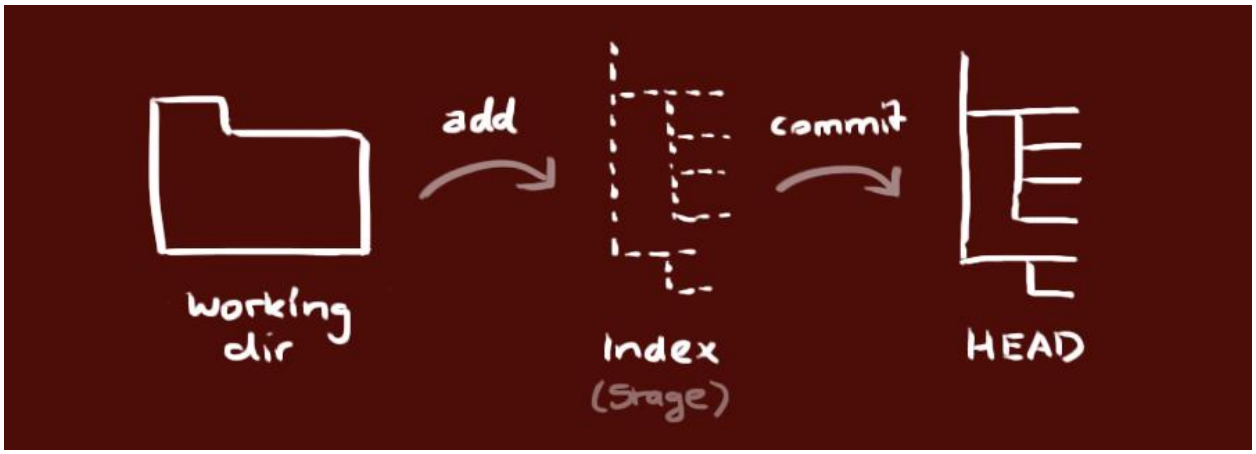
License: GPLv2, LGPLv2.1, and others

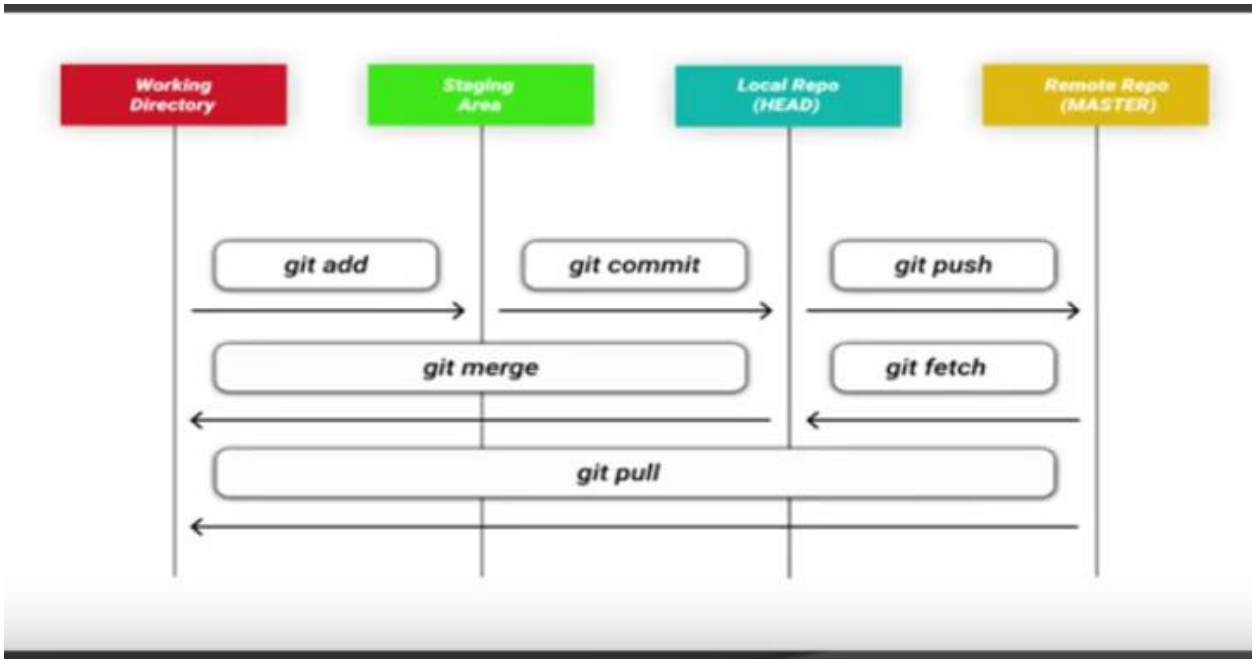
Written in: C, Perl, Tcl, Python

2. Alur-kerja

Repositori lokal Anda terdiri dari tiga bagian pokok yang disebut "trees" yang akan dikelola oleh git. Bagian yang pertama adalah `Direktori Kerja` yang menyimpan berkas aktual. `Indeks` berperan sebagai lokasi staging, yakni penyimpanan informasi tentang apa yang akan menjadi komit. `HEAD` munjukkan komit terakhir, yakni snapshot dari kumpulan file yang dikerjakan.

.





3. Perintah-Perintah Dasar Git

- *Git init* : untuk membuat repository pada file lokal yang nantinya ada folder .git
- *Git status* : untuk mengetahui status dari repository lokal
- *Git add* : menambahkan file baru pada repository yang dipilih

- *Git commit* : untuk menyimpan perubahan yang dilakukan, tetapi tidak ada perubahan pada remote repository.
- *Git push* : untuk mengirimkan perubahan file setelah di commit ke remote repository.
- *Git branch* : melihat seluruh branch yang ada pada repository
- *Git checkout* : menukar branch yang aktif dengan branchyang dipilih
- *Git merge* : untuk menggabungkan branch yang aktif dan branch yang dipilih
- *Git clone* : membuat Salinan repository local

4. Manfaat Git

- Dapat menyimpan seluruh versi source code.
- Dapat berkolaborasi dengan anggota tim dalam proyek.
- Dapat ikut berkontribusi ke poryek open-source.
- Lebih aman dan tertib digunakan untuk kolaborasi, karena kita dapat mengetahui apa yang diubah, kapan perubahannya dan siapa yang mengubahnya.
- Dapat memahami cara deploy aplikasi modern.
- Memudahkan revisi versi apabila terjadi kesalahan.

5. Contoh Software

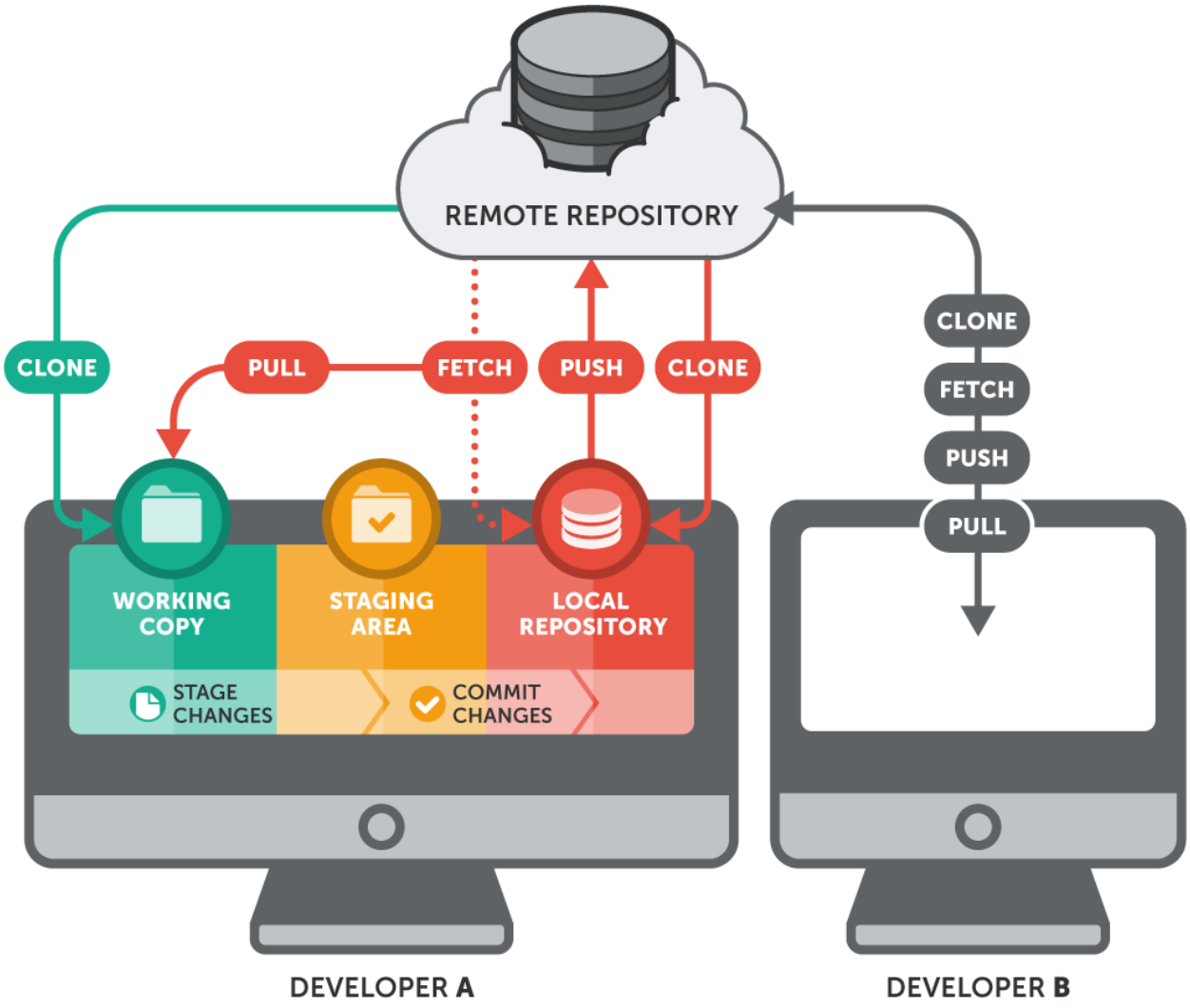
Contoh dari software version control system adalah github, bitbucket, gitlab, dan masih banyak lagi. Jika Anda sebagai developer belum mengetahui fitur git ini, maka Anda wajib mencoba dan memakainya. Karena banyak manfaat yang akan didapat dengan git ini.

- Github

- Bitbucket

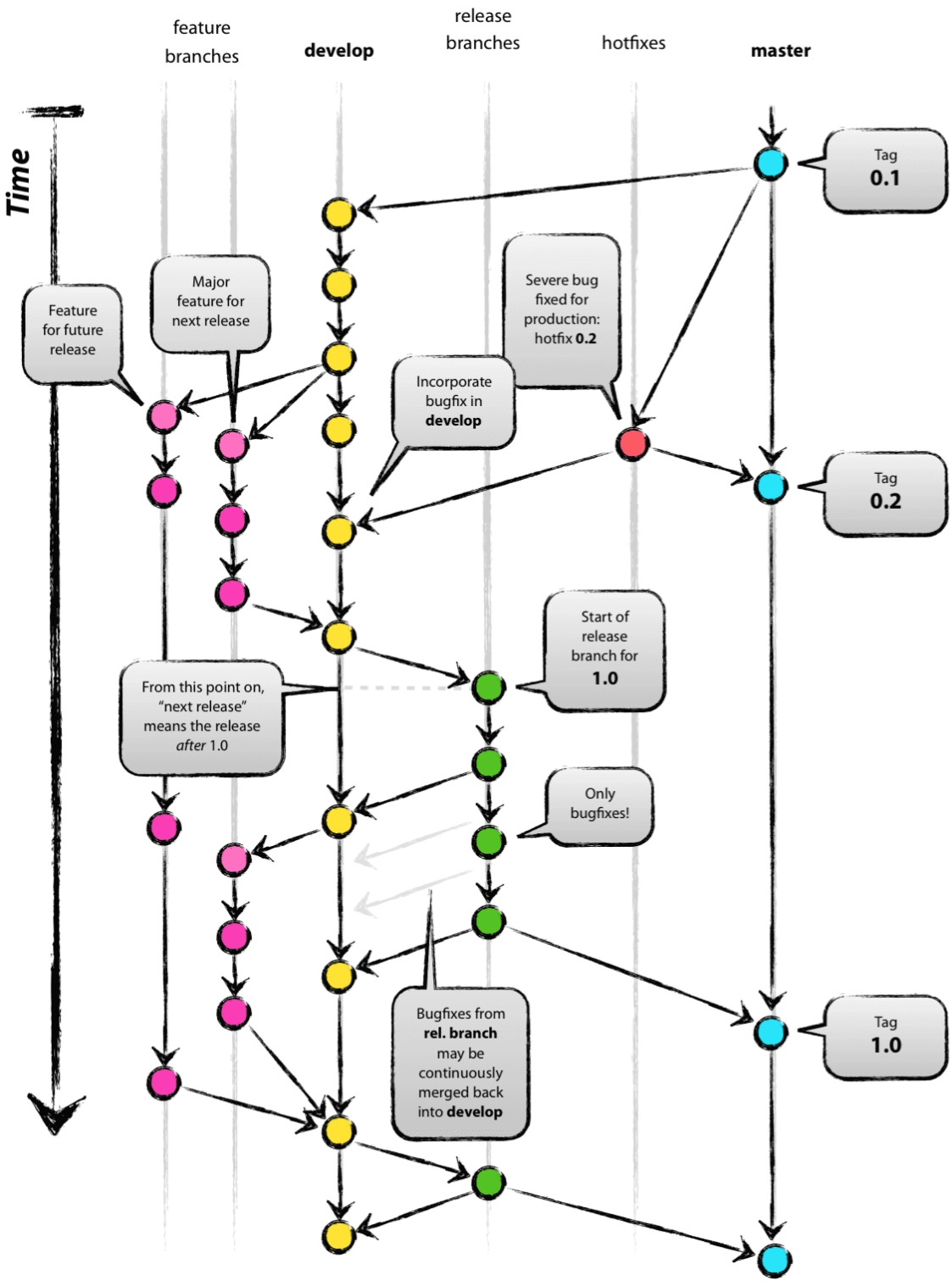
- Gitlab

6. **Alur Kerja Kolaborasi**



7. **GitFlow**

   Gitflow adalah mekanisme untuk mengelola percabangan Git yang diciptakan oleh Vincent Driessen.. Gitflow saat ini dianggap sebagai praktik terbaik untuk pengembangan perangkat lunak berkelanjutan modern dan praktik DevOps.


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Ada 5 branch yang biasa digunakan pada GitFlow :

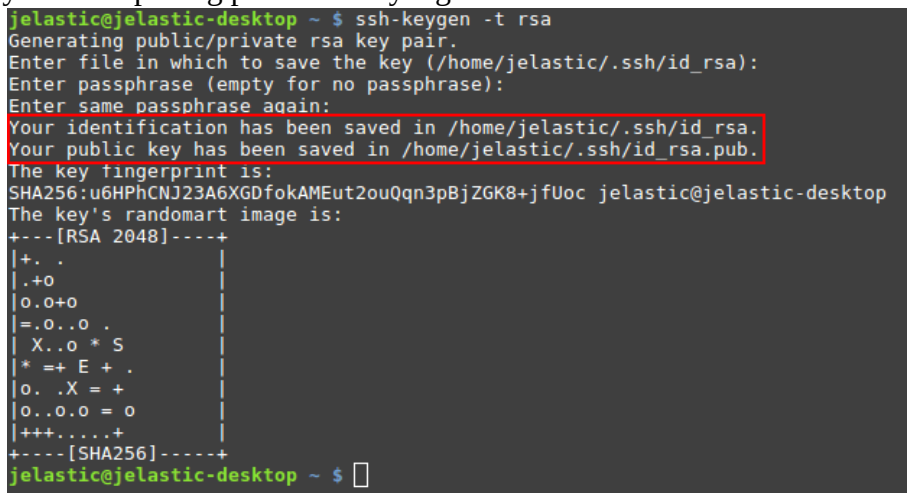
- Master
- Develop
- Feature
- Release
- Hotfix

Konsep :

- Develop branch dibuat dari main/master branch
- Release branch di buat dari develop branch
- Feature branch dibuat dari develop branch
- Ketika feature branch selesai dibuat, di merge Kembali ke develop branch
- Ketika release branch selesai dibuat, di merge Kembali ke develop dan main
- Ketika terdapat issue di main/master branch gunakan hotfix branch dari master/main branch untuk memperbaiki issue
- Ketika hotfix selesai, di merge Kembali ke master/main dan develop branch

8. SSH

SSH (Secure Shell) adalah aplikasi pengganti remote login seperti telnet, rsh, dan rlogin, yang jauh lebih aman. Fungsi utama aplikasi ini adalah untuk mengakses mesin secara remote. SSH Keys merupakan sebuah autentikasi ganda, diibaratkan sebuah kunci yang hanya bisa dipakai oleh identitas anda saja. SSH keys bekerja dengan sepasang kunci yaitu Public Key yang nantinya diletakkan pada sistem yang akan di remote, dan Private Key yang nantinya akan dipasang pada sistem yang meremote.

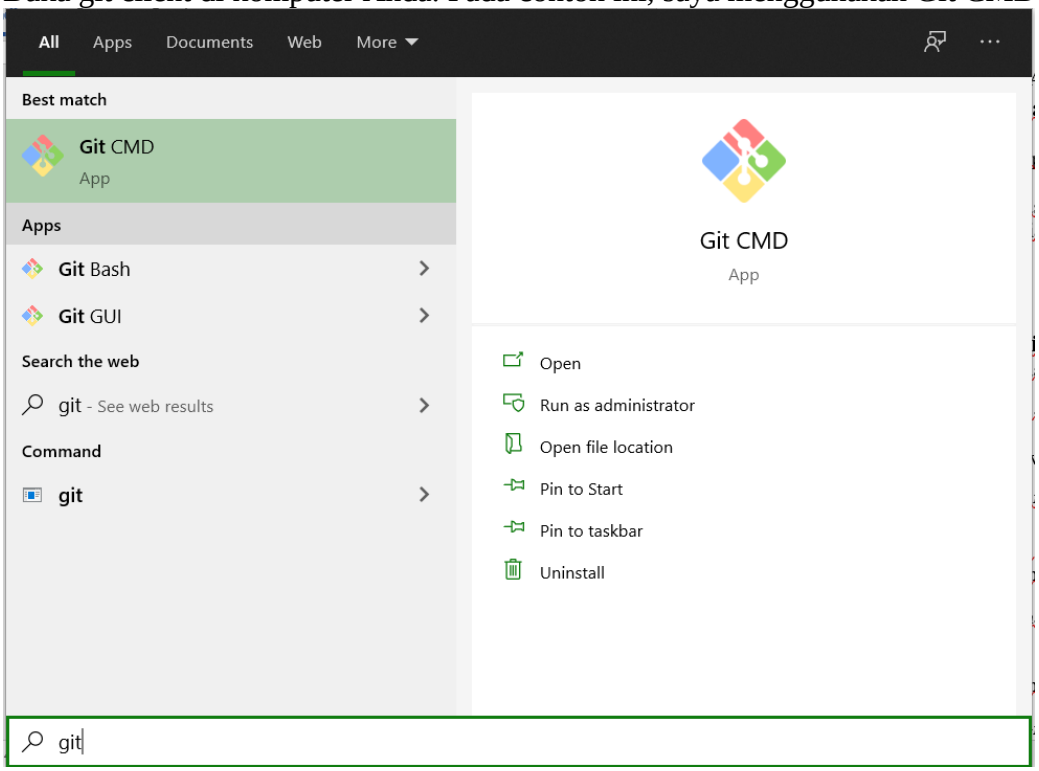


SSH memungkinkan kita untuk melakukan *push* ke repository github tanpa login. Berbeda dengan cara yang biasa (melalui HTTPS), kita harus memasukkan *username* dan *password* setiap kali melakukan *push*. Tapi dengan SSH kita tidak akan melakukan itu lagi.

D. PRAKTIKUM

1. Buat Repositori Baru

Buka git client di komputer Anda. Pada contoh ini, saya menggunakan Git CMD



Pada folder lokal, inisialisasikan git dengan cara menuliskan perintah:

```
git init
```

2. Periksa Repositori

buatlah salinan kerja dari repositori lokal dengan menjalankan perintah

```
git clone /jalur/ke/repositori
```

saat menggunakan server jarak-jauh, perintahnya menjadi

```
git clone namapengguna@host:/jalur/ke/repositori
```

3. Tambah & Komit

Anda dapat 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 "Pesan komit"
```

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

4. Mengirim Perubahan

Saat ini perubahan telah tersimpan di **HEAD** salinan kerja lokal Anda. Untuk mengirimkannya ke repositori jarak-jauh, lakukan

```
git push origin main
```

Ubah *master* sesuai cabang yang Anda inginkan.

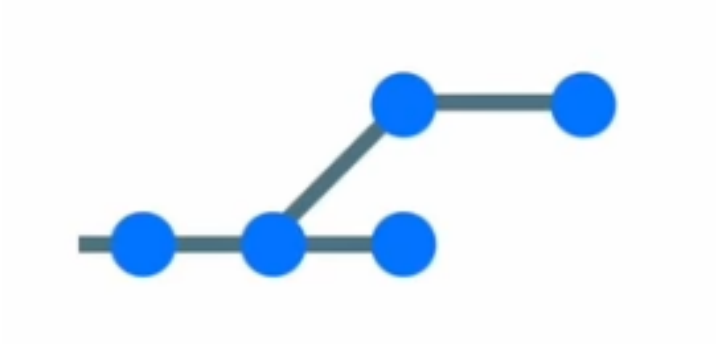
Jika repositori yang ada belum dikloning dan ingin dihubungkan ke server jarak-jauh, Anda perlu menambahkan

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

Sekarang Anda dapat mengirimkan perubahan ke server jarak-jauh yang dituju

5. Percabangan

Percabangan atau *branching* digunakan untuk mengembangkan fitur-fitur secara terisolasi. Cabang utama atau *master* merupakan cabang bawaan ketika Anda 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 main
```

dan hapus cabang yang tadi dibuat

```
git branch -d fitur_x
```

suatu cabang tidak terbuka untuk yang lainnya kecuali jika Anda mengirimkannya ke repositori jarak-jauh.

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

6. Perbaruan dan Penggabungan

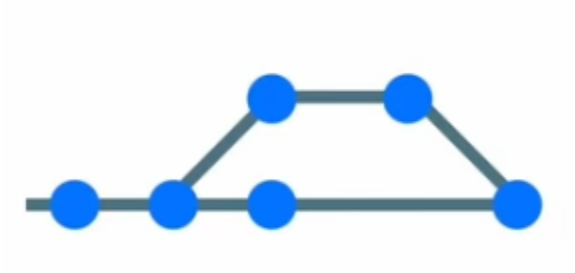
Untuk memperbarui repositori lokal ke komit terkini, lakukan

```
git pull
```

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

Untuk menggabungkan cabang lain ke cabang aktif (misal *master*), gunakan

```
git fetch
git merge <cabang>
```



Pada kasus diatas, git mencoba menggabungkan perubahan secara otomatis. Sayangnya hal ini tak selalu berjalan mulus dan Dapat menyebabkan *konflik*. Anda lah yang bertanggung jawab menggabungkan *konflik* tersebut secara manual dengan menyunting berkas yang ditunjukkan git. Setelah itu, Anda perlu memarkahinya dengan

```
git add <namaberkas>
```

sebelum penggabungan berlaku, Anda Dapat melakukan pratinjau menggunakan

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

Anda dapat mengatur ulang cabang Anda ke keadaan sebelum penggabungan jika Anda menemukan komit itu pada saat itu. Untuk memunculkan daftar semua HEAD yang kita miliki, kita dapat menggunakan.

```
git reflog
```

Untuk menampilkan data waktu dari tiap HEAD yang dimiliki, kita dapat menggunakan.

```
git reflog --relative-date
```

7. Menandai / Tagging

Sangat dianjurkan membuat penanda atau *tags* untuk perangkat lunak yang dirilis. Hal ini amat lah lazim, yang juga terjadi di SVN. Anda Dapat 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 Anda referensikan ke penanda.

8. Log

Dalam bentuknya yang paling sederhana, Anda dapat mempelajari riwayat repositori menggunakan

```
git log
```

Anda Dapat 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 Anda 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
```

9. Mengembalikan perubahan lokal

Seandainya Anda melakukan kesalahan, Anda Dapat mengembalikannya menggunakan perintah

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

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

10. Petunjuk lainnya

GUI git bawaan

```
gitk
```

menggunakan output git penuh warna

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

menunjukkan log satu baris per komit

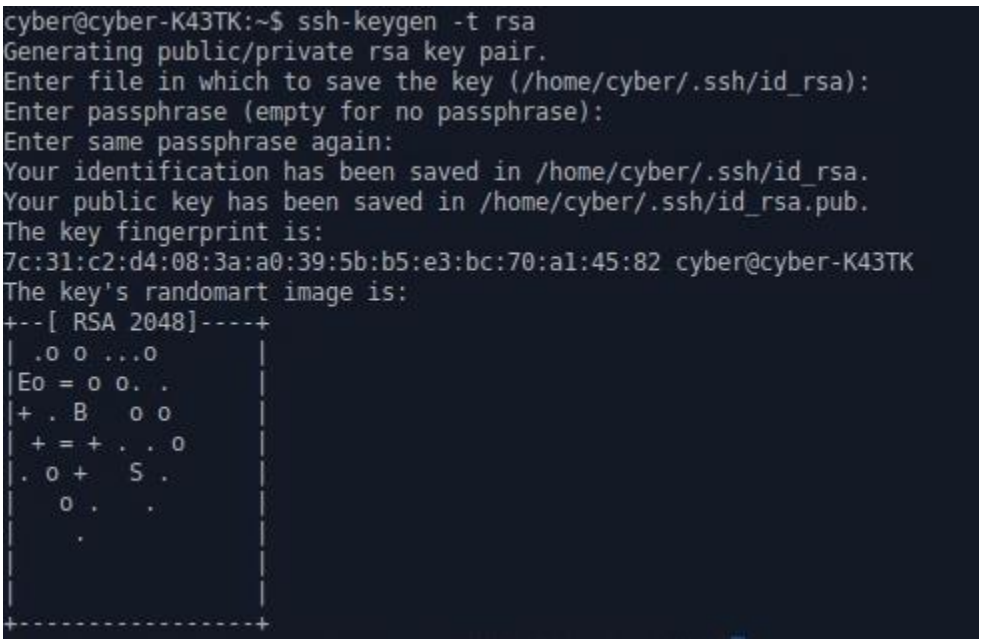
```
git config format.pretty oneline
```

menggunakan penambahan interaktif

```
git add -i
```

11.Tambahan : Membuat RSA Key

1. Sekarang kita harus membuat sepasang kunci yang nanti nya kita butuhkan untuk dipasang pada sistem. Pada panduan ini kami menggunakan sistem operasi Ubuntu 14.04 untuk membuat RSA Key. RSA Key pada sistem yang akan melakukan remote. Masukkan perintah berikut pada terminal: `ssh-keygen –t rsa`
2. Selanjutnya akan diinformasikan dimana file *Keys* nantinya akan disimpan, biasanya akan disimpan pada */home/user/.ssh/id_rsa*. Tekan Enter untuk melanjutkan.
3. Kemudian Masukkan *passphrase* , namun jika anda tidak ingin memberikan *passphrase* silahkan tekan Enter.
4. Selanjutnya akan di informasikan bahwa *Private Key* akan di simpan di */home/user/.ssh/id_rsa*, sedangkan *Public Key* disimpan pada /home/user/.ssh/id_rsa.pub dengan *fingerprint* sistem yang membuat RSA Key.
   Berikut contoh tampilannya :



5. Salin *Public Key* ke sistem yang akan diremote
6. Komponen yang kita butuhkan telah lengkap saat ini. kita telah memiliki *Public Key* dan *Private Key*, selanjutnya kita harus menyalin *Public Key* ke sistem yang akan kita remote agar sistem yang akan di remote mengenali sistem *client* kita.
   Gunakan perintah ssh-copy-id untuk menyalin *Public Key* ke sistem yang akan diremote.

   Kode : `ssh-copy-id user@ip_server_anda`

7. Lalu masukkan password dari user anda.
   Berikut tampilannya :

```
cyber@cyber-K43TK:~$ ssh-copy-id test@103.15.226.111
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
test@103.15.226.111's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'test@103.15.226.111'"
and check to make sure that only the key(s) you wanted were added.
```

8. Test merupakan nama user pada sistem yang akan diremote, untuk ip silahkan gunakan ip sistem yang akan di remote. Selain menggunakan perintah ssh-copy-id ada alaternatif lain yang dapat anda gunakan, anda dapat menyalin isi file id_rsa.pub kemudian anda *paste* pada sistem yang akan diremote di direktori */home/user/.ssh/authorized_keys*.

Sekarang jakuab sudah mengerti bagaimana Git bekerja.
Untuk praktikum ini, silakan kalian:

1. Mulai membiasakan diri dengan Git
2. Buat akun di web Github / Gitlab.
3. Buat remote repository yang berisi kumpulan karya kalian selama ini.

REFERENSI :

The Net Ninja Tutorial
https://www.youtube.com/watch?v=3RjQznt-8kE&list=PL4cUxeGkcC9goXbgTDQ0n_4TBzOO0ocPR

IDStack Tutorial
https://www.youtube.com/watch?v=i7fnAxHAp0M&list=PL1aMeb5UP_PHXTV_Xpt-19x_rVPXrymOM

GitFlow
https://medium.com/easyread/gitflow-in-practice-1d8a2bbdc3a1