GitHub
Git Training

Hari Pertama

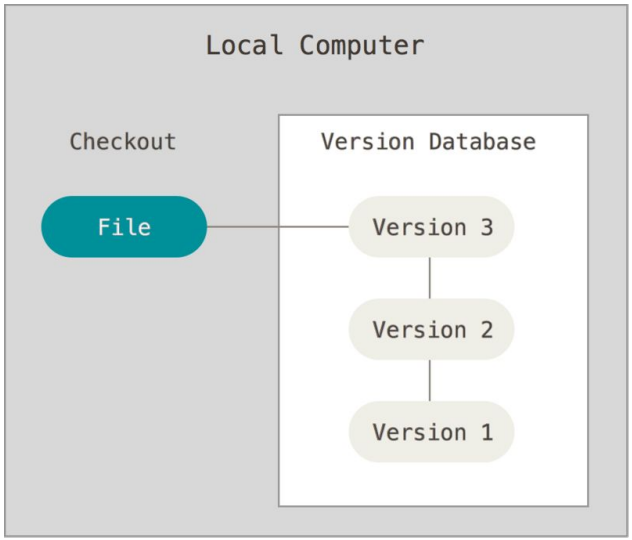
Bermulanya dengan Version Control

Version Control merupakan suatu sistem yang merekodkan perubahan yang dibuat kepada suatu fail atau set fail dari semasa ke semasa supaya anda boleh mengembalikan versi tertentu pada masa hadapan.

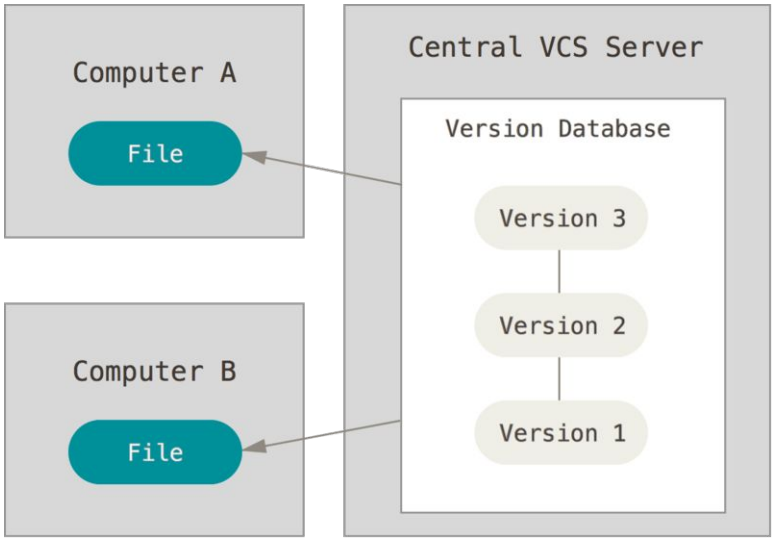
Jenis Version Control :-

1. **Local** Version Control Systems
2. **Centralized** Version Control Systems
    - CVS, Subversion, and Perforce
3. **Distributed** Version Control Systems
    - Git, Mercurial, Bazaar or Darcs

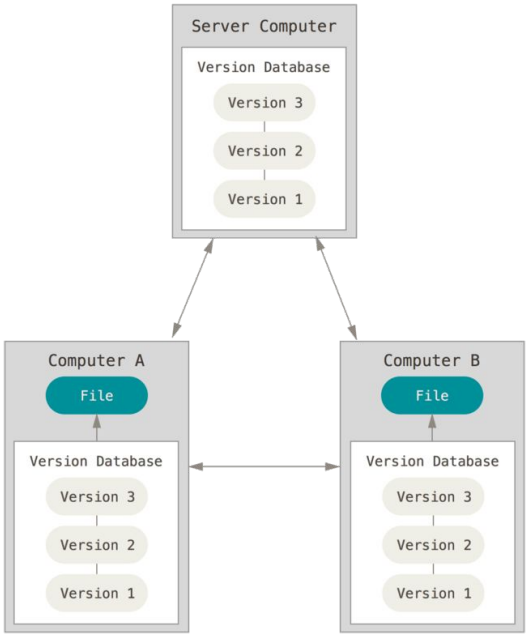
Local Version Control Systems



Centralized Version Control Systems



Distributed Version Control Systems



Sejarah Pendek bagi Git

Tahun 2002 projek kernel Linux menggunakan DVCS yang dikenali sebagai BitKeeper.

Tahun 2005 kerjasama antara komuniti Linux dan syarikat BitKeeper dibatalkan.

Mendorong komuniti Linux untuk membangunkan DVCS sendiri.

Tahun 2005 Git dilancarkan dan semakin berkembang.

Asas Git

Cara kerja Git berbeza dengan CVS, Subversion atau Perforce.

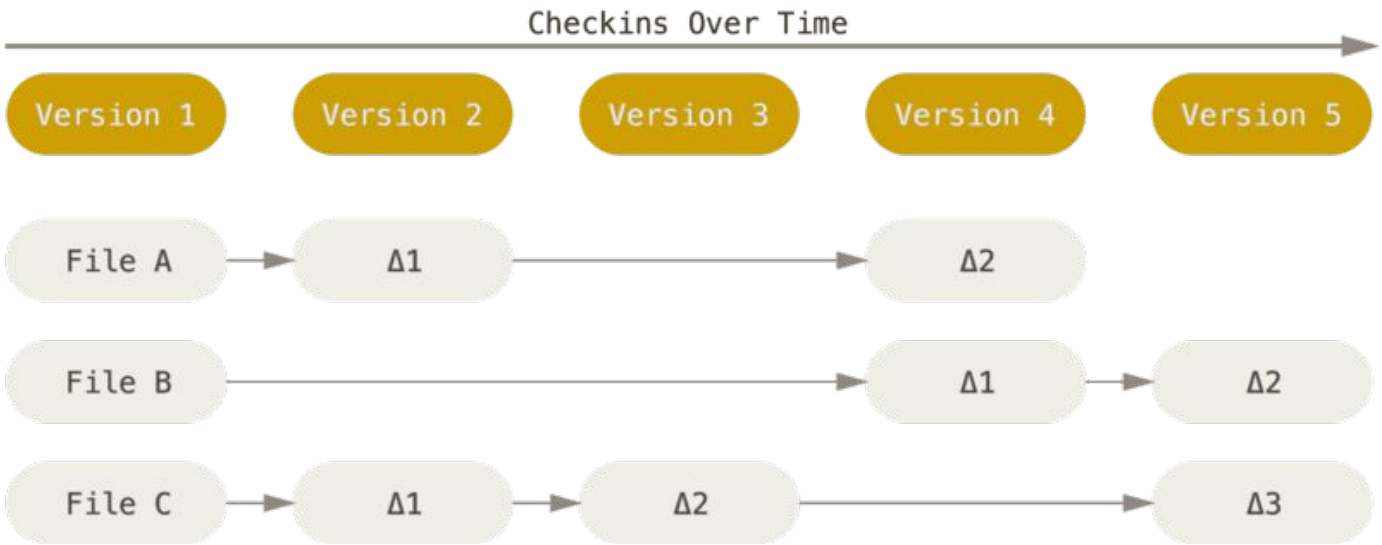
Git mampu menyimpan dan memproses maklumat dengan cara yang sangat berbeza.

Kebanyakan sistem lain menyimpan maklumat sebagai suatu senarai mengenai perubahan berasaskan fail.

Sistem lain (CVS, Subversion, Perforce, Bazaar, dan sebagainya) memproses maklumat yang disimpan sebagai satu set fail.

Perubahan yang dibuat adalah kepada setiap fail.

Asas Git


Checkins Over Time
Version 1
Version 2
Version 3
Version 4
Version 5
File A
Δ1
Δ2
File B
Δ1
Δ2
File C
Δ1
Δ2
Δ3

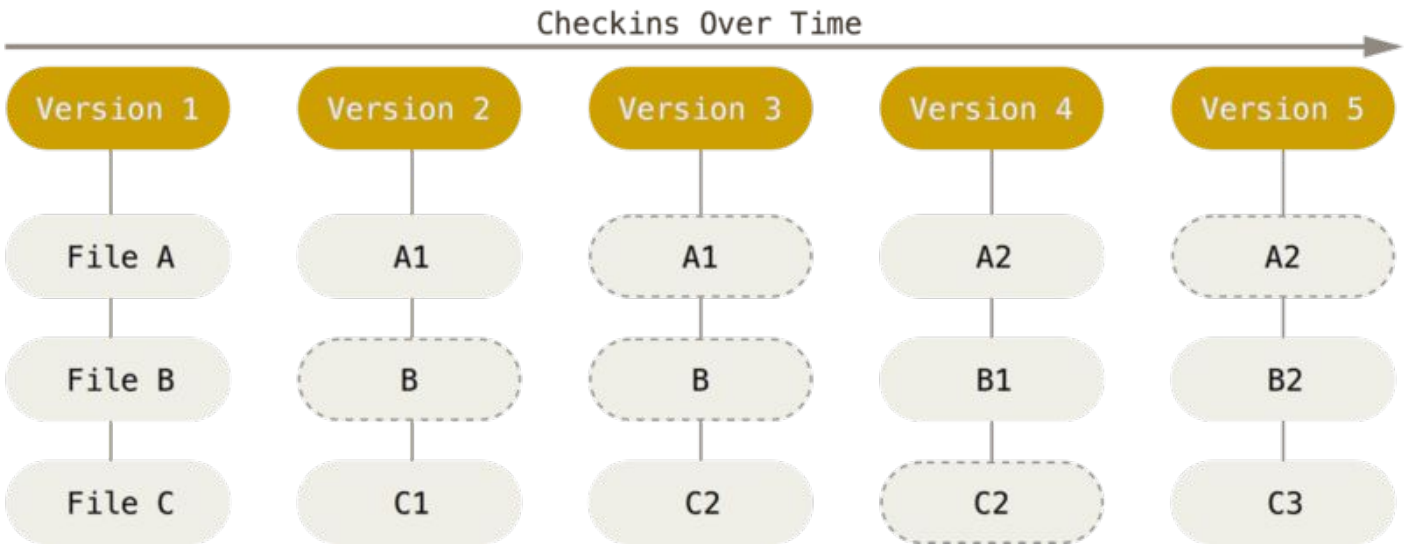
Asas Git

Git memproses datanya seperti suatu siri snapshot dari sebuah filesystem kecil.

Setiap kali melakukan commit atau save file dari projek, Git mengambil sebuah gambar tentang bagaimana tampilan semua file pada saat itu dan menyimpannya.

jika file-file tersebut tidak berubah, Git tidak menyimpan filenya lagi, tetapi hanya menautkan ke file yang sama seperti sebelumnya yang telah tersimpan.

Asas Git


Checkins Over Time
Version 1
Version 2
Version 3
Version 4
Version 5
File A
File B
File C
A1
B
C1
A1
B
C2
A2
B1
C2
A2
B2
C3

Setiap Operasi Adalah Bersifat Local

Kebanyakan operasi di Git hanya memerlukan fail dan sumber local untuk beroperasi.

Git tidak perlu pergi ke server untuk mendapatkan history dan menampilkannya, ia membacanya langsung dari lokal.

Git dapat mencari file sebulan yang lalu dan melakukan perbandingan secara lokal, bukannya dengan meminta kepada remote server.

Git Mempunyai Integriti

Semuanya dalam Git telah dilakukan checksum sebelum disimpan dan kemudian merujuk pada checksum tersebut.

Git melakukan checksum dengan SHA-1 hash (kumpulan kata sepanjang 40 karakter).

Git menyimpan semua informasi dalam databasenya bukan dari nama file, tetapi dari nilai hash isinya.

Ketika melakukan sesuatu tindakan dalam Git, hampir semuanya hanya menambahkan data ke database Git.

Tiga Keadaan Git

Git memiliki tiga keadaan utama yang file-file dapat masuk ke dalamnya: modified, staged dan committed.

1. Dimodifikasikan ataupun **modified** bermaksud fail telah diubah suai tetapi belum melakukan komit ke dalam database local lagi.
2. Ditandakan ataupun **staged** bermaksud sesuatu fail yang diubah suai telah ditandakan untuk seterusnya masuk ke dalam komit.
3. Dikomit ataupun **committed** bermaksud bahawa data telah disimpan dengan selamat dalam database local.

Tiga bahagian utama Git

Ini membawa kepada tiga bahagian utama bagi sesuatu projek Git

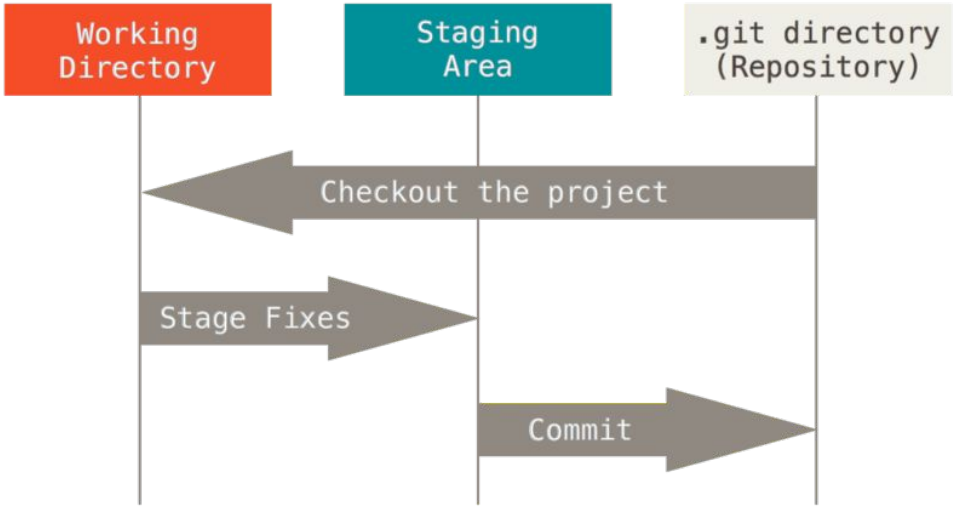
1. **Direktori Git** (repository)) - Tempat pemenyimpan metadata dan database.
2. **Working tree** - Terdiri dari file-file yang sedang dikerjakan.
3. **Staging area** - Menyimpan maklumat tentang apa yang akan masuk ke dalam komit seterusnya.

Aliran kerja Git adalah seperti berikut:

1. Anda mengubah suai file dalam working tree.
2. Anda menyiapkan filenya dan menambah snapshot ke staging area.
3. Anda melakukan commit.

Pengubah suaian yang dilakukan di working tree tidak akan di commit secara terus ke repository, tetapi ianya ditandakan dahulu di staging area.

Tiga bahagian utama Git



Instalasi & Pengaturan Git

1. Pemasangan pada Windows
   - http://git-scm.com/download/win
2. Pengaturan pengenal
   - git config --global user.name "John Doe"
   - git config --global user.email johndoe@example.com
3. Pengaturan editor
   - git config --global core.editor notepad
4. Memeriksa Pengaturan
   - git config --list
5. Mendapatkan Bantuan
   - git help <verb>
   - git <verb> --help

Mendaftar GitHub

1. Membuat account github
    - https://github.com/

Mendapatkan Repository Git

Lazimnya, project Git boleh didapatkan menggunakan dua pendekatan utama:

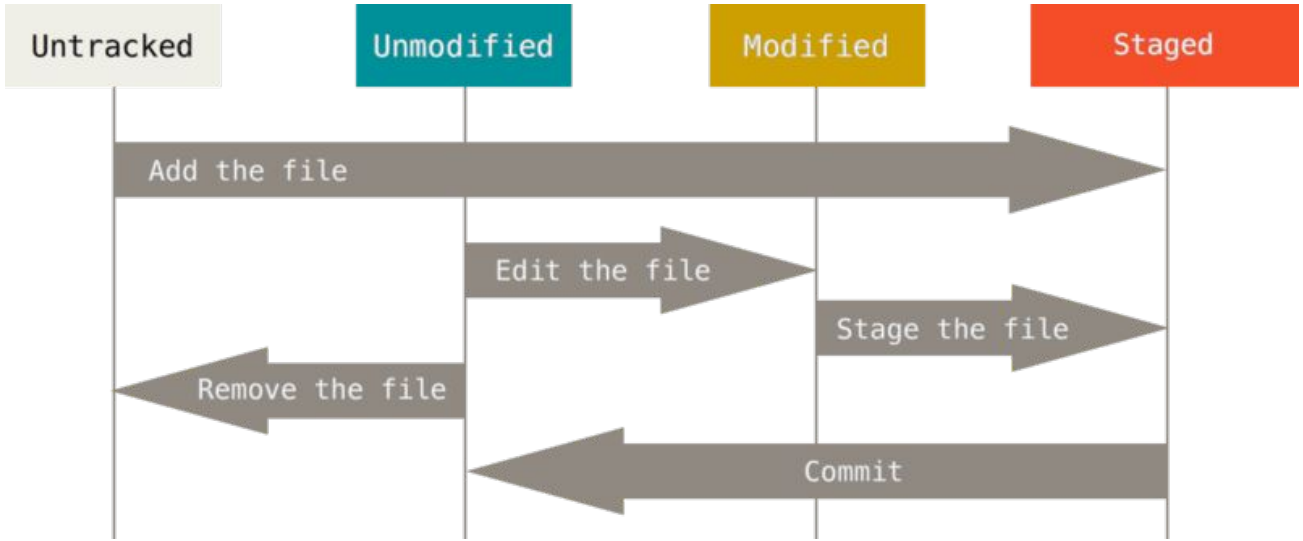
1. Mengambil direktori local yang tidak berada di bawah kawalan versi, dan mengubahnya menjadi suatu repositori Git.
    - cd /c/user/my_project
    - git init
2. Mengeklonkan repositori Git yang sedia ada dari server.
    - git clone https://github.com/repo/repo

Perubahan pada Repositori

Setiap file di dalam working directory berada di 2 keadaan

1. Tracked
    - File yang berada dalam snapshot terakhir; dapat berada dalam keadaan unmodified, modified, atau staged
2. Untracked
    - File yang dalam working directory yang tidak berada dalam snapshot terakhir dan juga tidak berada di area staging
    - Kebiasaannya terjadi ketika pertama kali create sebuah repositori atau menambah file baru

Perubahan pada Repositori



Syntax pada Git

1. Memeriksa status fail
    a. git status
2. Memantau file baru & file yang Modified
    a. git add <FILE>
3. Mengecualikan file
    a. Tambahkan parameter pada file .gitignore
    b. Standard glob patterns dapat digunakan
    c. Baris kosong atau dimula dengan # akan diabaikan
    d. Mengakhiri pola dengan slash (/) untuk direktori
    e. Menafikan pola dengan menggunakan tanda seru (!) pada permulaan

Syntax pada Git

```
# ignore all .a files
*.a

# but do track lib.a, even though you're ignoring .a files above
!lib.a

# only ignore the TODO file in the current directory, not subdir/TODO
/TODO

# ignore all files in the build/ directory
build/

# ignore doc/notes.txt, but not doc/server/arch.txt
doc/*.txt

# ignore all .pdf files in the doc/ directory and any of its subdirectories
doc/**/*.pdf
```

Syntax pada Git

1. Melihat perubahan dalam stage dan luar stage
    - git diff
2. Commit perubahan
    - git commit
    - git commit -m "<MESSAGE>"
3. Memadam file
    - git rm <FILE>
4. Memindahkan file
    - git mv <FROM> <TO>

Syntax pada Git

1. Melihat history commit
    - git log
      git log --stat
      git log --pretty=oneline
2. Mengubah/membatalkan commit terakhir
    - git commit --amend
3. Unstaging a Staged File
    - git reset HEAD <FILE>
4. Unmodifying a Modified File
    - git checkout -- <FILE>

Syntax pada Git

1. Melihat daftar tag
    - git tag
    - git show <TAG>
      git show v1.5
2. Membuat tag asas
    - git tag v1.4.0
3. Membuat tag dengan keterangan
    - git tag -a v1.4 -m 'version 1.4'

Bekerja dengan remote repository

Repositori remotes adalah sekumpulan versi dari project yang disiarkan di Internet atau di jaringan.

1. Melihat remote repository
    - git remote
      git remote -v
2. Menambah remote repository
    - git remote add <REMOTE> <URL>
    - git remove add origin https://github.com/Daklyq/git-training.git
3. Mengambil dan menarik dari remote repository
    - git fetch <REMOTE>
      git pull <REMOTE> <BRANCH>
      git pull origin master

Bekerja dengan remote

1. Menolak ke remote repository
   - git push <REMOTE> <BRANCH>
     git push origin master
2. Memeriksa remote repository
   - git remote show <REMOTE>
3. Mengganti nama remote repository
   - git remote rename <OLD_REMOTE> <NEW_REMOTE>
     git remote rename origin git
4. Memadam remote repository
   - git remote rm <REMOTE>

Tamat