

«SKRIPSI/TUGAS AKHIR»

«JUDUL BAHASA INDONESIA»



«Nama Lengkap»

NPM: «10 digit NPM UNPAR»

PROGRAM STUDI «MATEMATIKA/FISIKA/TEKNIK INFORMATIKA»  
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS  
UNIVERSITAS KATOLIK PARAHYANGAN  
«tahun»



«FINAL PROJECT/UNDERGRADUATE THESIS»

«JUDUL BAHASA INGGRIS»



«Nama Lengkap»

NPM: «10 digit NPM UNPAR»

DEPARTMENT OF «MATHEMATICS/PHYSICS/INFORMATICS»  
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES  
PARAHYANGAN CATHOLIC UNIVERSITY  
«tahun»



# LEMBAR PENGESAHAN

«JUDUL BAHASA INDONESIA»

«Nama Lengkap»

NPM: «10 digit NPM UNPAR»

Bandung, «tanggal» «bulan» «tahun»

Menyetujui,

Pembimbing Utama

Pembimbing Pendamping

«pembimbing utama/1»

«pembimbing pendamping/2»

Ketua Tim Penguji

Anggota Tim Penguji

«penguji 1»

«penguji 2»

Mengetahui,

Ketua Program Studi

«Ketua Program Studi»



## PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa «skripsi/tugas akhir» dengan judul:

**«JUDUL BAHASA INDONESIA»**

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,  
Tanggal «tanggal» «bulan» «tahun»

Meterai Rp. 6000
---------------------

«Nama Lengkap»  
NPM: «10 digit NPM UNPAR»





## ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

**Kata-kata kunci:** «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»



## ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

**Keywords:** «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»



*«kepada siapa anda mempersembahkan skripsi ini...?»*



## KATA PENGANTAR

«Tuliskan kata pengantar dari anda di sini ...»

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Bandung, «bulan» «tahun»

Penulis





## DAFTAR ISI

<b>KATA PENGANTAR</b>	<b>xv</b>
<b>DAFTAR ISI</b>	<b>xvii</b>
<b>DAFTAR GAMBAR</b>	<b>xix</b>
<b>DAFTAR TABEL</b>	<b>xxi</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	1
1.3 Tujuan . . . . .	1
1.4 Batasan Masalah . . . . .	2
1.5 Metodologi . . . . .	2
1.6 Sistematika Pembahasan . . . . .	2
<b>2 LANDASAN TEORI</b>	<b>3</b>
2.1 Git . . . . .	3
2.1.1 Version Control Systems . . . . .	3
2.1.2 Cara Kerja Git . . . . .	5
2.1.3 Operasi Dasar pada Git . . . . .	8
2.1.4 Git Checkout . . . . .	9
2.2 JGit . . . . .	10
2.2.1 Porcelain API . . . . .	10
2.2.2 Plumbing API . . . . .	10
2.3 Selenium WebDriver . . . . .	10
2.4 Apache Commons CLI . . . . .	10
<b>DAFTAR REFERENSI</b>	<b>11</b>
<b>A KODE PROGRAM</b>	<b>13</b>
<b>B HASIL EKSPERIMEN</b>	<b>15</b>



## DAFTAR GAMBAR

2.1	Local version control . . . . .	3
2.2	Centralized version control . . . . .	4
2.3	Distributed version control . . . . .	5
2.4	Menyimpan data sebagai <i>snapshots</i> dari <i>project</i> . . . . .	6
2.5	Menyimpan data sebagai perubahan terhadap versi dasar dari setiap <i>file</i> . . . . .	6
2.6	<i>Working tree</i> , <i>Staging area</i> , dan Git direktori . . . . .	7
2.7	Contoh histori <i>commit</i> dalam pengembangan perangkat lunak . . . . .	7
2.8	<i>Checkout</i> pada <i>commit</i> . . . . .	9
B.1	Hasil 1 . . . . .	15
B.2	Hasil 2 . . . . .	15
B.3	Hasil 3 . . . . .	15
B.4	Hasil 4 . . . . .	15



## DAFTAR TABEL



# BAB 1

## PENDAHULUAN

### 1.1 Latar Belakang

Git merupakan perangkat lunak *Version Control Systems*[1]. *Version control* adalah sistem yang merekam perubahan pada *file* atau sekumpulan *file* dari waktu ke waktu. Perubahan yang terjadi pada *repository* dicatat oleh Git dalam bentuk histori *commit*. Setiap *commit* mengandung informasi mengenai perubahan yang terjadi pada *repository*, waktu perubahan, dan orang yang melakukan perubahan. *Database* pada *git* tidak bersifat terpusat, melainkan terdistribusi. Setiap orang yang terlibat mempunyai *database* lokal pada masing-masing komputer, sehingga pengelolaan perangkat lunak dapat dilakukan secara *online* dan *offline*.

JGit adalah *library* Java murni yang mengimplementasikan Git *version control systems*[2]. JGit dikembangkan oleh Eclipse Foundation. JGit bersifat *open source*. Dengan menggunakan JGit, fitur-fitur dalam Git dapat diakses melalui program Java.

Selenium adalah seperangkat alat yang secara khusus digunakan untuk mengotomatisasi *web browsers*[3]. Dengan menggunakan Selenium WebDriver, pengguna dapat memasukkan *script* bahasa pemrograman tertentu untuk melakukan pengujian. Bahasa pemrograman yang didukung yaitu C#, Java, Perl, PHP, Python, Ruby, dan JavaScript. Selenium WebDriver dapat melakukan pengujian pada *browser* Google Chrome, Mozilla Firefox, Opera, Safari, dan Internet Explorer.

Pada skripsi ini, akan dibuat sebuah perangkat lunak yang dapat menampilkan animasi *time-lapse* dari pengembangan proyek perangkat lunak berbasis web. Perangkat lunak ini dibangun menggunakan bahasa Java. Perangkat lunak ini menggunakan tampilan terminal/konsol. Dalam pembuatan animasi *timelapse*, dibutuhkan perangkat lunak Selenium WebDriver dan JGit.

### 1.2 Rumusan Masalah

Rumusan masalah dari skripsi ini adalah sebagai berikut:

1. Bagaimana cara membangkitkan animasi *timelapse* pada pengembangan proyek perangkat lunak berbasis web?
2. Bagaimana cara mengimplementasikan aplikasi untuk membangkitkan *timelapse* pada pengembangan proyek perangkat lunak berbasis web?

### 1.3 Tujuan

Tujuan dari skripsi ini adalah sebagai berikut:

1. Mengetahui cara untuk membangkitkan animasi *timelapse* pada pengembangan proyek perangkat lunak berbasis web.
2. Mengetahui cara untuk mengimplementasikan aplikasi untuk membangkitkan *timelapse* pada pengembangan proyek perangkat lunak berbasis web.

## 1.4 Batasan Masalah

Batasan masalah pada skripsi ini adalah sebagai berikut:

1. Perangkat lunak ini hanya membangkitkan animasi *timelapse* untuk perangkat lunak berbasis web.
2. Masukan perangkat lunak berupa alamat direktori proyek perangkat lunak yang terekam oleh Git.

## 1.5 Metodologi

Metodologi penelitian yang digunakan dalam skripsi ini adalah sebagai berikut:

1. Melakukan studi literatur tentang Git, Selenium WebDriver, Git, dan JGit.
2. Melakukan analisis penggunaan Selenium WebDriver dan JGit untuk membangkitkan animasi *timelapse*.
3. Merancang perangkat lunak.
4. Membangun perangkat lunak.
5. Melakukan eksperimen dan pengujian pada perangkat lunak.

## 1.6 Sistematika Pembahasan

Setiap bab dalam penelitian ini memiliki sistematika penulisan yang dijelaskan ke dalam poin-poin sebagai berikut:

1. Bab 1: Pendahuluan, yaitu membahas mengenai gambaran umum penelitian ini. Berisi tentang latar belakang, rumusan masalah, tujuan, batasan masalah, metode penelitian, dan sistematika penulisan.
2. Bab 2: Dasar Teori, yaitu membahas mengenai teori-teori yang mendukung berjalannya penelitian ini. Berisi tentang teori Git, JGit, Selenium WebDriver, dan Apache Commons CLI.
3. Bab 3: Analisis, yaitu membahas mengenai analisa masalah. Berisi tentang analisis penggunaan Jgit dan Selenium WebDriver untuk membangkitkan animasi *timelapse*.
4. Bab 4: Perancangan, yaitu membahas mengenai perancangan yang dilakukan sebelum melakukan tahapan implementasi. Berisi tentang perancangan perangkat lunak pembangkit *timelapse* proyek pengembangan perangkat lunak.
5. Bab 5: Implementasi dan Pengujian, yaitu membahas mengenai implementasi dan pengujian aplikasi yang telah dilakukan. Berisi tentang implementasi dan hasil pengujian aplikasi.
6. Bab 6: Kesimpulan dan Saran, yaitu membahas hasil kesimpulan dari keseluruhan penelitian ini dan saran-saran yang dapat diberikan untuk penelitian berikutnya.



## BAB 2

### LANDASAN TEORI

Pada bab ini dibahas dasar teori yang mendukung berjalannya skripsi ini. Dasar teori yang dibahas yaitu Git, JGit, Selenium WebDriver, dan Apache Commons CLI.

#### 2.1 Git

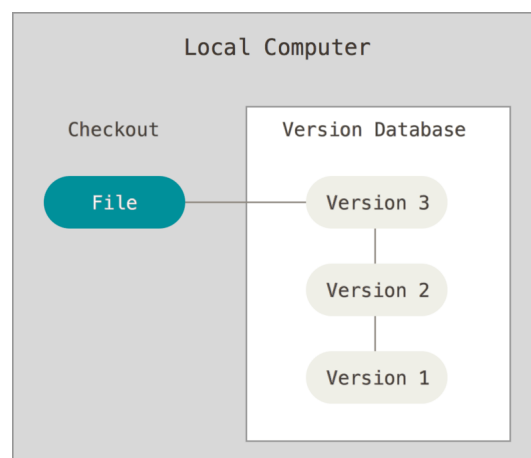
Seperti yang telah dijelaskan pada subbab 1.1, Git merupakan perangkat lunak *Version Control Systems*. Pada subbab ini, dijelaskan mengenai *Version Control Systems*, cara kerja Git, *Git checkout*, dan operasi-operasi dasar pada Git.

##### 2.1.1 Version Control Systems

*Version Control Systems* adalah sistem yang merekam perubahan pada *file* atau sekumpulan *file* dari waktu ke waktu[1]. *Version Control Systems* biasanya digunakan untuk merekam *file* yang berisi *source code program*, tetapi pada kenyataannya *Version Control Systems* dapat merekam hampir semua jenis *file* dalam komputer. Terdapat tiga jenis *Version Control Systems*, yaitu: *local Version Control Systems*, *centralized Version Control Systems*, dan *distributed Version Control Systems*.

##### Local Version Control Systems

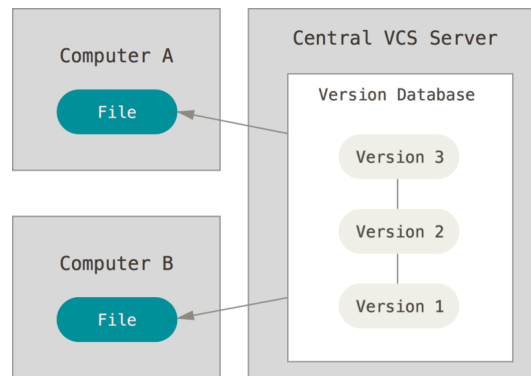
Metode *version-controlled* yang banyak digunakan orang adalah dengan cara menyalin sekumpulan *file* ke direktori lain[1]. Namun cara tersebut rentan terhadap *error*. Misalnya, terdapat direktori A dan B, pengguna ingin mengubah *file* yang terdapat pada direktori B, tetapi pengguna lupa kalau dia sedang berada di direktori A, maka pengguna mengubah *file* pada direktori yang salah. Untuk mengatasi masalah tersebut, *programmer* mengembangkan *local Version Control Systems*.



Gambar 2.1: Local version control

Gambar 2.1 merupakan struktur dari *Local Version Control Systems*. *Database local Version Control Systems* ini tersimpan pada *local* direktori di komputer. *Database* ini menyimpan perubahan *file* ke dalam beberapa versi atau *state*. *Local Version Control*, dapat melakukan *checkout file* ke versi atau *state* tertentu.

### Centralized Version Control Systems



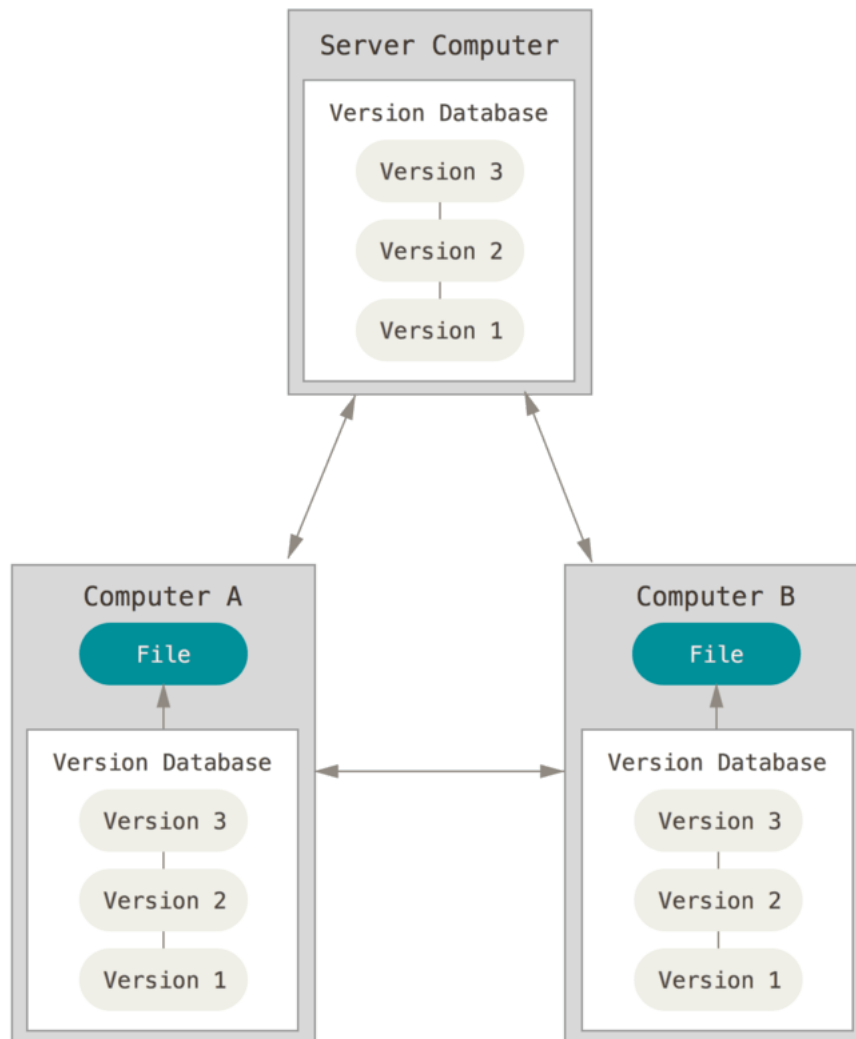
Gambar 2.2: Centralized version control

*Local Version Control* hanya menyimpan *file* pada satu komputer saja. Muncul masalah baru ketika *user* ingin berkolaborasi dengan *user* lain. Untuk mengatasi masalah ini dikembangkan *Centralized version control*. Gambar 2.2 merupakan struktur dari *Centralized Version Control Systems*. Dalam *Centralized Control Version Systems* terdapat sebuah *server* yang menyimpan setiap versi *file*, dan klien yang dapat melakukan *checkout file*<sup>[1]</sup>.

Sistem *Centralized Version Control Systems* memiliki beberapa kelebihan. Setiap *user* dapat mengetahui pekerjaan yang dilakukan oleh *user* lain. Administrator dapat lebih mudah mengontrol *database Centralized Version Control Systems* dibandingkan dengan *database Local Version Control Systems* dari setiap klien.

Tetapi, *Centralized Version Control Systems* juga memiliki kelemahan. Jika *server* pusat *Centralized Version Control Systems* mati, maka perubahan pada *file* tidak bisa disimpan. Klien juga tidak dapat melakukan kolaborasi dengan klien lain. Jika *harddisk* pada server rusak, maka semua versi *file* akan hilang.

## Distributed Version Control Systems

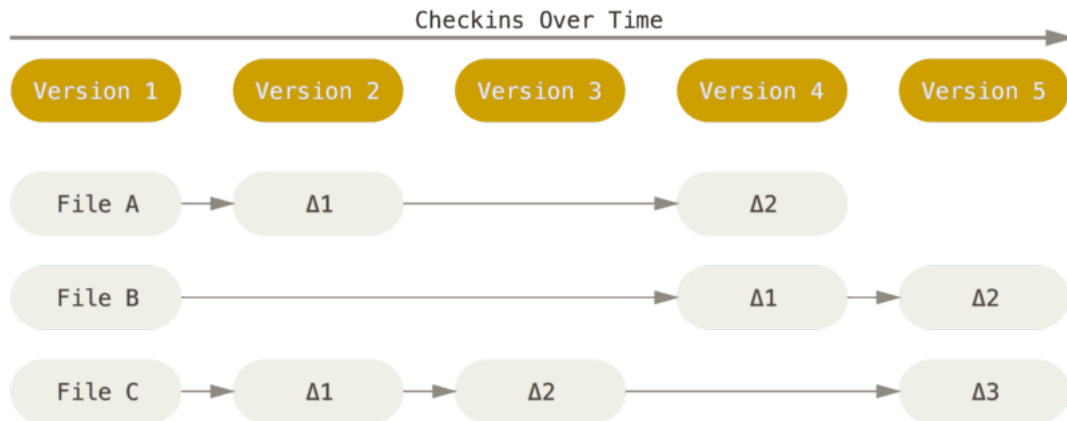


Gambar 2.3: Distributed version control

Gambar 2.3 merupakan struktur dari *Distributed Version Control Systems*. Dalam sebuah DVCS (seperti Git, Mercurial, Bazaar atau Darcs), klien tidak hanya melakukan *checkout* untuk *snapshot* terakhir setiap *file*, namun klien juga memiliki salinan dari repositori tersebut[1]. Dengan kata lain setiap klien memiliki *version database local* pada komputernya. Jika server pusat mati, klien masih bisa melakukan kolaborasi dan klien manapun dapat mengirimkan kembali salinan repositori ke *server*.

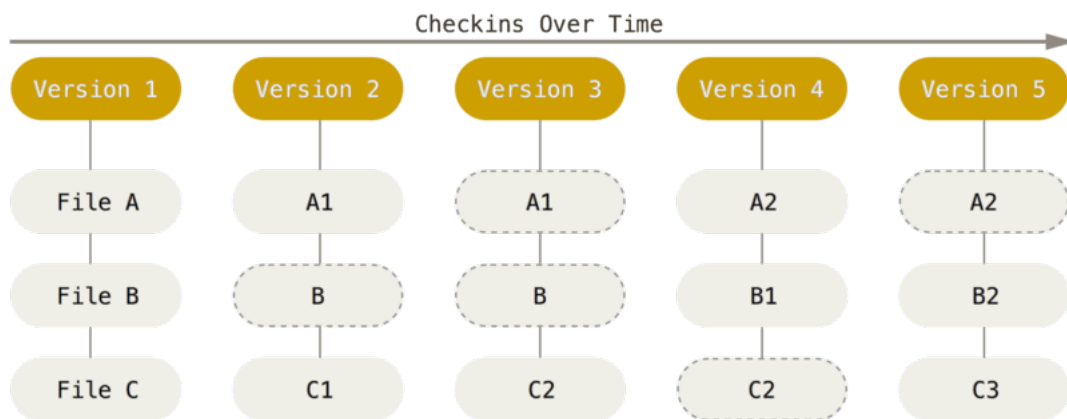
### 2.1.2 Cara Kerja Git

Salah satu perbedaan antara Git dengan VCS lainnya adalah dalam cara Git memperlakukan datanya[1]. Kebanyakan sistem *Version Control Systems* lain menyimpan informasi sebagai daftar perubahan *file*. Pada Gambar 2.4, terdapat tiga *file*. *Version Control Systems* menyimpan *file* A, B, dan C pada versi pertama saja. Untuk versi kedua dan seterusnya yang disimpan adalah perubahan pada setiap *file*. Sistem ini disebut juga sebagai *delta-based Version Control Systems*.



Gambar 2.4: Menyimpan data sebagai *snapshots* dari *project*

Berbeda dengan *Version Control Systems* lainnya, Git memperlakukan datanya sebagai sebuah kumpulan *snapshot* dari sebuah miniatur *file system*[1]. Setiap kali dilakukan *commit*, git merekam *state* dari sekumpulan *file* dan menyimpanannya sebagai *reference snapshot* tersebut. Gambar 2.5, menunjukkan *snapshots* dari *file* A, B, dan C. Pada versi kedua, *file* B tidak mengalami perubahan, sehingga *file* yang disimpan adalah referensi *file* B pada versi sebelumnya.



Gambar 2.5: Menyimpan data sebagai perubahan terhadap versi dasar dari setiap *file*

### State pada Git

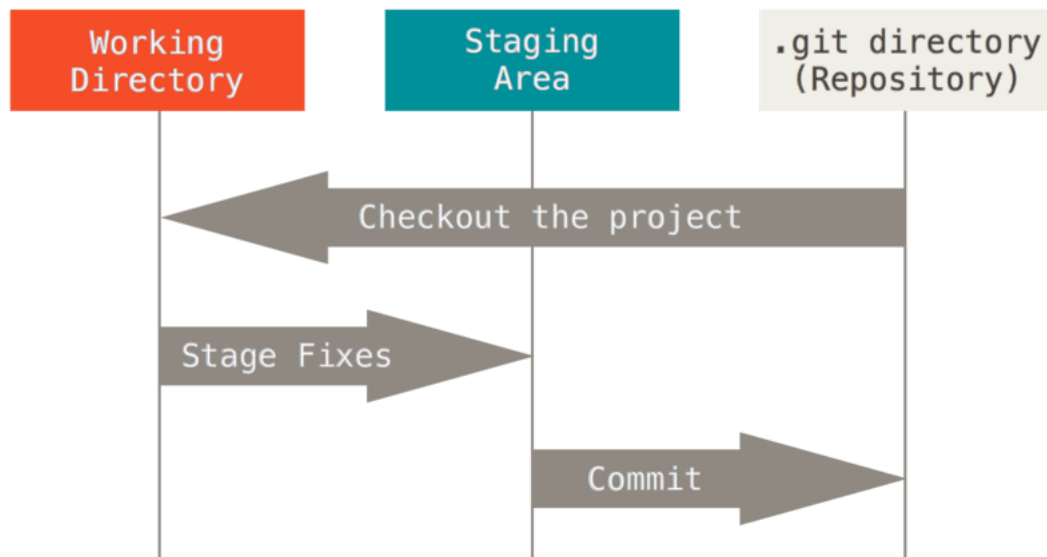
Terdapat tiga *state* pada Git yaitu *committed*, *modified*, and *staged*[1]. *Committed* adalah *state* dimana data sudah disimpan di *local database*. *Modified state* dimana terdapat perubahan pada *file*, namun *file* tersebut belum di *commit* ke *database*. *Staged* adalah *state* dimana *file* telah ditandai untuk kemudian dilakukan *commit*.

Terdapat tiga bagian utama dari sebuah *project* Git yaitu direktori Git, direktori kerja, dan staging area[1]. Direktori Git merupakan tempat dimana Git menyimpan *metadata* dan *object database* dari *project*. *Working tree* adalah suatu *snapshot* dari *project*. Sekumpulan *file* ini diambil dari *database* di direktori Git dan ditempatkan pada *disk* untuk digunakan dan dimodifikasi. *Staging area* adalah *file* yang menyimpan informasi mengenai apa yang menjadi *commit* selanjutnya. *File staging area* terdapat pada direktori Git. Untuk lebih jelasnya, lihat Gambar 2.6.

Alur kerja dari Git adalah sebagai berikut:

1. Melakukan modifikasi pada *file*
2. Menandai perubahan pada *file* dan memindahkannya ke *staging area*.

3. Mengambil *file* dari *staging area* dan menyimpan *snapshot* ke direktori Git. Proses ini disebut dengan *commit*.



Gambar 2.6: *Working tree*, *Staging area*, dan Git direktori

## Commit

Commit merupakan sebuah *snapshot* dari suatu *file* atau direktori. *Commit* menggambarkan *state* dari *working directory*. Gambar 2.5 menunjukkan terdapat tiga *file* pada versi/*commit* keempat. Dimana terdapat *file* A1, B1, dan C1 pada *working directory*. *File* A1, B1, dan C2 merupakan *state file* A, B, dan C pada *commit* keempat.

Git melakukan *check-summed* pada *commit* sebelum menyimpannya ke Git repositori. Mekanisme yang digunakan untuk melakukan *check-summed* disebut dengan *SHA-1 hash*[1]. *SHA-1 hash* terdiri dari empat puluh karakter heksadesimal(0-9 a-f). Nilai dari *SHA-1 hash* dihitung berdasarkan isi dari *working directory* atau struktur direktori Git.

```

Command Prompt - git log
C:\Users\user\Documents\GitHub>cd train-tracker-ellena-angelica
C:\Users\user\Documents\GitHub\train-tracker-ellena-angelica>git log
commit b8aeacbd4743619b7b2d790d45bde26b899641e0 (HEAD -> master, origin/master, origin/HEAD)
Author: adamadamadamadamadam <adamnurmishwari@gmail.com>
Date: Thu May 3 01:15:31 2018 +0700

    commitan terakhir. mastiin g buang memory sm batre

commit f836cc65bf6d50e274df54aa06c6fb529667aa06
Author: Evelyn Wijaya <evelynwijaya777@gmail.com>
Date: Wed May 2 22:03:25 2018 +0700

    Update README.md

commit 2e1ce9a03a1f417326c3c6586503303cf6daf6b8
Author: Evelyn Wijaya <evelynwijaya777@gmail.com>
Date: Wed May 2 22:01:10 2018 +0700

    Create README.md

commit 2f04488f9008745e8e6f67da33ffb2f6c2c9e747
Author: Evelyn Wijaya <evelynwijaya777@gmail.com>
Date: Wed May 2 14:08:02 2018 +0700

    fix stasiun double
  
```

Gambar 2.7: Contoh histori *commit* dalam pengembangan perangkat lunak

Seperti yang diperlihatkan pada Gambar 2.7, setiap *commit* memiliki beberapa informasi. Baris pertama menunjukkan *commit* ID yang berupa *SHA-1 hash*. Pada baris ini, *Master* menunjukkan *branch* yang sedang aktif, *master* juga merupakan *pointer* ke *commit* terakhir. *Head* merupakan *reference* ke *branch master*. *Origin/master* dan *origin/HEAD* merupakan *master* dan *HEAD* pada *remote repository*. Baris kedua menunjukkan orang yang melakukan *commit* dan alamat emailnya. Baris ketiga menunjukkan waktu terjadinya *commit*. Baris terakhir berisi deskripsi dari *commit* tersebut.

### 2.1.3 Operasi Dasar pada Git

Pada subbab ini dijelaskan mengenai operasi dasar dalam Git dan sintaks-sintaksnya. Sintaks-sintaksnya ini dimasukkan pada Git *command line*. Berikut ini adalah operasi-operasi dasar dalam Git:

1. Init

Operasi ini digunakan untuk membuat repositori lokal baru dengan nama tertentu. Bisa juga digunakan untuk merekam direktori yang sudah ada. Berikut adalah sintaks untuk melakukan operasi *init*:

```
$ git init [project-name]
```

2. Add

Operasi ini digunakan untuk menandai perubahan pada *file* dan memindahkan *file* tersebut ke *staging area*. Operasi ini juga digunakan untuk menambahkan *file* yang dipantau perubahannya. Berikut adalah sintaks untuk melakukan operasi *add*:

```
$ git add [file]
```

3. Commit

Operasi digunakan untuk merekam *snapshot* atau *state file* atau sekumpulan *file*. Operasi ini juga digunakan untuk memindahkan *file* yang berada di *staging area* ke repositori Git. Berikut adalah sintaks untuk melakukan operasi *commit*:

```
$ git commit -m "[descriptive message]"
```

4. Branch

Operasi ini digunakan untuk menampilkan semua *branch* yang ada pada repositori Git, membuat *branch* baru, dan menghapus *branch*. Berikut adalah sintaks-sintaks untuk melakukan operasi *branch*:

```
$ git branch
```

```
$ git branch [branch-name]
```

```
$ git branch -d [branch-name]
```

```
$ git branch -D [branch-name]
```

5. Diff

Operasi ini digunakan untuk menampilkan perbedaan pada *file* yang belum masuk *staging area*, menampilkan perbedaan pada *file* yang berada di *staging area* dengan *file* di *commit* sebelumnya, dan perbedaan *file* antara dua *branch*. Berikut adalah sintaks-sintaks untuk melakukan operasi *diff*:

```
$ git diff
```

```
$ git diff --staged
```

```
$ git diff [first-branch]...[second-branch]
```

6. Clone

Operasi ini digunakan untuk menyalin repositori Git yang berada di komputer lain atau suatu *server*. Berikut adalah sintaks untuk melakukan operasi *clone*:

```
$ git clone [url]
```

## 7. Fetch

Operasi ini digunakan untuk mengambil data dari *remote* repositori ke repositori lokal. Berikut adalah sintaks untuk melakukan operasi *fetch*:

```
$ git fetch [bookmark]
```

## 8. Merge

Operasi ini digunakan untuk menggabungkan *branch* tertentu dengan *branch* yang sedang aktif. Operasi ini juga digunakan untuk menggabungkan data yang diambil dari *remote* repositori dengan data pada *working directory*. Berikut adalah sintaks untuk melakukan operasi *merge*:

```
$ git merge [branch]/[bookmark]
```

## 9. Pull

Operasi ini adalah gabungan dari operasi *fetch* dan *merge*. Berikut adalah sintaks untuk melakukan operasi *pull*:

```
$ git pull
```

## 10. Push

Operasi ini digunakan untuk mengirim data pada repositori Git lokal ke *remote repository*. Berikut adalah sintaks untuk melakukan operasi *push*:

```
$ git push [alias] [branch]
```

## 11. Checkout

Operasi ini digunakan untuk berpindah ke *branch* atau *commit* tertentu, setelah itu memperbarui *file* pada *working directory* berdasarkan *branch* atau *commit* tersebut. Berikut ini adalah sintaks-sintaks untuk operasi *checkout*:

```
$ git checkout [SHA-1 commit]
```

```
$ git checkout [branch-name]
```

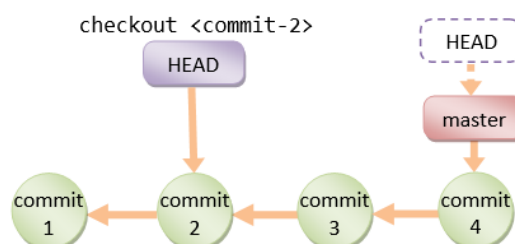
## 12. Log

Operasi ini digunakan untuk menampilkan semua histori *commit* pada *branch* yang sedang aktif. Berikut ini adalah sintaks untuk melakukan operasi *log*:

```
$ git log
```

### 2.1.4 Git Checkout

Seperti yang sudah dijelaskan pada subbab 2.1.3, *checkout* dapat digunakan untuk ke *branch* atau *commit* tertentu. Operasi *checkout* dapat dilakukan menggunakan sintaks "\$ git checkout" diikuti dengan nama *branch* atau *SHA-1 hash*. Gambar 2.8 menunjukkan contoh *checkout* pada *commit*. Posisi awal *HEAD* menunjuk pada *branch master*, setelah dilakukan *checkout* ke *commit* kedua, posisi *HEAD* menunjuk pada *commit* kedua. *Working directory* diperbarui berdasarkan *state* pada *commit* kedua.



Gambar 2.8: *Checkout* pada *commit*

## **2.2 JGit**

### **2.2.1 Porcelain API**

### **2.2.2 Plumbing API**

## **2.3 Selenium WebDriver**

## **2.4 Apache Commons CLI**



## DAFTAR REFERENSI

- [1] Chacon, S. dan Straub, B. (2014) *Pro Git* The expert's voice. Apress.
- [2] Jgit | the eclipse foundation. <https://www.eclipse.org/jgit/>. [Online; diakses 2-September-2018].
- [3] Selenium webdriver. <https://www.seleniumhq.org/about/>. [Online; diakses 2-September-2018].



# LAMPIRAN A

## KODE PROGRAM

Listing A.1: MyCode.c

```

1 // This does not make algorithmic sense,
2 // but it shows off significant programming characters.
3
4 #include<stdio.h>
5
6 void myFunction( int input, float* output ) {
7     switch ( array[i] ) {
8         case 1: // This is silly code
9             if ( a >= 0 || b <= 3 && c != x )
10                 *output += 0.005 + 20050;
11             char = 'g';
12             b = 2^n + ~right_size - leftSize * MAX_SIZE;
13             c = (--aaa + &daa) / (bbb++ - ccc % 2 );
14             strcpy(a,"hello_$@?");
15         }
16         count = ~mask | 0x00FF00AA;
17     }
18 }
19
20 // Fonts for Displaying Program Code in LATEX
21 // Adrian P. Robson, nepsweb.co.uk
22 // 8 October 2012
23 // http://nepsweb.co.uk/docs/progfonts.pdf

```

Listing A.2: MyCode.java

```

1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.HashSet;
4
5 //class for set of vertices close to furthest edge
6 public class MyFurSet {
7     protected int id; //id of the set
8     protected MyEdge FurthestEdge; //the furthest edge
9     protected HashSet<MyVertex> set; //set of vertices close to furthest edge
10    protected ArrayList<ArrayList<Integer>> ordered; //list of all vertices in the set for each trajectory
11    protected ArrayList<Integer> closeID; //store the ID of all vertices
12    protected ArrayList<Double> closeDist; //store the distance of all vertices
13    protected int totaltrj; //total trajectories in the set
14
15    /*
16     * Constructor
17     * @param id : id of the set
18     * @param totaltrj : total number of trajectories in the set
19     * @param FurthestEdge : the furthest edge
20     */
21    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
22        this.id = id;
23        this.totaltrj = totaltrj;
24        this.FurthestEdge = FurthestEdge;
25        set = new HashSet<MyVertex>();
26        ordered = new ArrayList<ArrayList<Integer>>();
27        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
28        closeID = new ArrayList<Integer>(totaltrj);
29        closeDist = new ArrayList<Double>(totaltrj);
30        for (int i = 0;i <totaltrj;i++) {
31            closeID.add(-1);
32            closeDist.add(Double.MAX_VALUE);
33        }
34    }
35
36 }

```



## LAMPIRAN B

### HASIL EKSPERIMEN

Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4