Merancang *Git Server* dengan Pendekatan *GitHub Social Coding* dalam Peningkatan Pembelajaran Mahasiswa

SKRIPSI

Diajukan Untuk Memenuhi Sebagian Syarat Guna
Memperoleh Gelar Sarjana Komputer (S.Kom.)
Pada Program Studi Teknik Informatika

OLEH:

M. SAIFUL MUKHAROM
NPM. 09.1.03.02.0299

FAKULTAS TEKNIK
UNIVERSITAS NUSANTARA PERSATUAN GURU REPUBLIK INDONESIA
UNP KEDIRI
2015

Skripsi oleh :

M. SAIFUL MUKHAROM
NPM. 09.1. 03.02.0299

Judul :

Merancang *Git Server* dengan Pendekatan *GitHub Social Coding* dalam Peningkatan Pembelajaran Mahasiswa

Telah disetujui untuk dilanjutkan Kepada
Panitia Ujian/Sidang Skripsi Program Studi Teknik Informatika
Fakultas Teknik UNP Kediri

Tanggal : 23 Mei 2015

Pembimbing I	Pembimbing II
Dr. M. Muchson, SE., M.M	**Ari Eka Prasetiyanto, S.Kom**
NIDN. 0018126701	NIDN. 0729048403

Skripsi oleh:

M. SAIFUL MUKHAROM
NPM. 09.1.03.02.0299

Judul:

Merancang *Git Server* dengan Pendekatan *GitHub Social Coding* dalam Peningkatan Pembelajaran Mahasiswa

Telah dipertahankan di depan Panitia Ujian/Sidang Skripsi
Program Studi Teknik Informatika
Fakultas Teknik UNP Kediri
Pada Tanggal: 23 Mei 2015

Dan Dinyatakan telah Memenuhi Persyaratan

Panitia Penguji :

1. Ketua : Rini Indriati, S.Kom., M.Kom.
2. Penguji I : Nursalim, S.Pd., MH.
3. Penguji II : Patmi Kasih, M.Kom.

Mengetahui,
Dekan Fakultas Teknik

Rini Indriati, S.Kom., M.Kom.
NIY. 1081001070

PERNYATAAN

Yang bertanda tangan di bawah ini saya,

Nama	: M. Saiful Mukharom
Jenis Kelamin	: Laki–laki
Tempat/tgl. Lahir	: Nganjuk/ 06 Januari 1990
NPM	: 09.1.03.02.0299
Fak/Jur/Prodi	: Fakultas Teknik/S1 Teknik Informatika

menyatakan dengan sebenarnya, bahwa dalam Skripsi ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu perguruan tinggi, dan sepanjang pengetahuan saya tidak terdapat karya tulis atau pendapat yang pernah diterbitkan oleh orang lain, kecuali yang secara sengaja dan tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Kediri, 23 Mei 2015
Yang Menyatakan

M. SAIFUL MUKHAROM
NPM. 09.1.03.02.0299

MOTTO DAN PERSEMBAHAN

Motto:

Pelajarilah dasarnya ilmu,

atau engkau tidak akan dapat memahami ilmu dengan luas.

[Ungkapan Ulama]

Adab-adabnya ilmu itu lebih banyak daripada ilmu itu sendiri.

[Abu Abdillah al-Balkhi rahimahullah]

Karya ini buat:

Seluruh Keluargaku Tercinta dan Para Penuntut Ilmu.

Abstrak

M. Saiful Mukharom: Merancang *Git Server* dengan Pendekatan *GitHub Social Coding* dalam Peningkatan Pembelajaran Mahasiswa, Skripsi, Teknik Informatika, FT UNP kediri, 2015.

Kata kunci: *git server, github, social coding,* peningkatan pembelajaran.

Selama ini *Git Server* hanya dimanfaatkan sebagai pengembangan aplikasi, belum banyak yang menerapkan sebagai media pembelajaran, sehingga banyak ditemukan media belajar mahasiswa yang masih personal (individu). *Git Server* adalah layanan *Source Code Management* (SCM) dalam jaringan yang terdistribusi. *Git Server* dengan pendekatan *GitHub Social Coding* dirancang untuk meningkatkan pembelajaran mahasiswa dalam mata kuliah pemrograman. *Social Coding* adalah suatu konsep mendistribusikan dan mengkolaborasikan *repository* secara *open source* maupun *close source*.

Git Server dirancang dengan layanan *Domain Name System* (DNS) *server*, *Dynamic Host Configuration Protocol* (DHCP) *server*, akses protokol SSH dan HTTP, *gitolite* sebagai manajemen *repository* dan *user*, serta *git* sebagai layanan *bare repository*, rancangan layanan tersebut untuk mendapatkan *url* SSH: *git@git.gitserver:path/to/repo_name.git* dan *url* HTTP: *http://www.gitserver/*. Alamat *url* sebagai akses kolaborasi *repository* yang *read-write* maupun *read-only*.

Rancangan *Git Server* setelah diujikan kepada mahasiswa dengan tahapan pengujian terbatas pertama terukur 87,79% kategori sangat layak, pengujian terbatas kedua terukur 85,70% kategori sangat layak, pengujian luas terukur 73,33% kategori layak, dengan kesimpulan bahwa untuk merancang *Git server* dengan pendekatan *GitHub Social Coding* harus dapat digunakan sebagai *public repository* maupun *private repository,* kemudian dapat memanajemen *repository* dan *users,* serta dapat digunakan sebagai kolaborasi belajar bersama mata kuliah pemrograman yang terdistribusi. Dengan demikian diimplikasikan bahwa Rancangan *git server* dengan pendekatan *github social coding* terbukti dapat digunakan sebagai media belajar mata kulih pemrograman serta terbukti dapat meningkatkan pembelajaran mata kuliah pemrograman, yaitu dengan peningkatan pembelajaran mahasiswa terukur 66,53% mahasiswa lebih aktif dalam pembelajaran, 80,00% mahasiswa menginginkan mengasah kemampuan pemrogramannya dengan mahasiswa lain, 81,63% mahasiswa ingin merefleksikan apa dan bagaimana mereka belajar.

KATA PENGANTAR

Sesungguhnya segala puji hanya milik Allah Rabb semesta alam, hanya kepada-Nya kami memuji, meminta pertolongan, dan memohon ampunan. Kami berlindung kepada Allah dari semua kejelekan jiwa dan keburukan perbuatan kami. Barang siapa yang diberi petunjuk oleh Allah maka tiada yang mampu menyesatkannya dan barang siapa yang disesatkan oleh Allah maka tidak akan ada yang bisa memberikan hidayah kepadanya. Aku bersaksi bahwasanya tidak ada illah yang berhak diibadahi selain Allah Yang Maha Esa, tidak ada sekutu bagi-Nya dan aku bersaksi bahwasanya Muhammad adalah hamba dan Rasul-Nya.

Puji Syukur Kami panjatkan kehadirat Allah Tuhan Yang Maha Kuasa, karena hanya atas perkenan-Nya tugas penyusunan skripsi ini dapat terselesaikan.

Skripsi dengan judul "**Merancang *Git Server* dengan Pendekatan *GitHub Social Coding* dalam Peningkatan Pembelajaran Mahasiswa**" ini ditulis guna memenuhi sebagian syarat untuk memperoleh gelar Sarjana Komputer, pada Jurusan Teknik Informatika Fakultas Teknik Universitas Nusantara PGRI Kediri.

Pada kesempatan ini diucapkan terimakasih dan penghargaan yang tulus kepada :

1. Orang Tua (Ibu) & Keluarga yang telah memberikan kesempatan, kepercayaan, dukungan, dan doa selama ini untuk selalu belajar dan berjuang untuk masa depan.

2. Dr. H. Samari, S.E., MM. Rektor UNP Kediri yang memberikan dorongan motivasi kepada mahasiswa.
3. Rini Indriati, S. Kom., M. Kom. Dekan FT UNP Kediri yang telah memberikan dorongan dan masukkan yang membangun kepada peneliti.
4. Ahmad Bagus Setiawan, S.T., MM., M.Kom. Ketua Program Studi Teknik Informatika yang telah memberikan banyak himbauan kepada peneliti.
5. Bagus Fadzerie Robby, S.Kom. selaku Dosen Pembimbing yang telah membantu banyak dalam penyusunan proposal.
6. Ibu Resty Wulaningrum, M.Kom. atas masukkan, kritikan dan sarannya pada sidang proposal yang begitu berguna dan membuka wawasan peneliti dalam perbaikan proposal.
7. Ari Eka Prasetiyanto, S.Kom. selaku Dosen Pembimbing yang telah membantu banyak dalam penyusunan skripsi.
8. Dr. M. Muchson, SE., MM. selaku Dosen Pembimbing yang telah membantu banyak dalam penyusunan skripsi.
9. Artinawati Dwi Lestari, S.P., M.Pd. selaku dosen mata kuliah yang telah membantu dan memberikan dorongan serta motivasi dalam penyusunan skripsi.
10. Teman–teman Komunitas CAH UNP yang telah membantu banyak dalam penyusunan skripsi.
11. Ucapan terima kasih juga disampaikan kepada pihak – pihak lain yang tidak dapat disebutkan satu persatu, yang telah banyak membantu menyelesaikan skripsi ini.

Disadari bahwa skripsi ini masih banyak kekurangan, maka diharapkan, kritik, dan saran dari berbagai pihak sangat diharapkan.

Kediri,

M. SAIFUL MUKHAROM
NPM. 09.1.03.02.0299

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

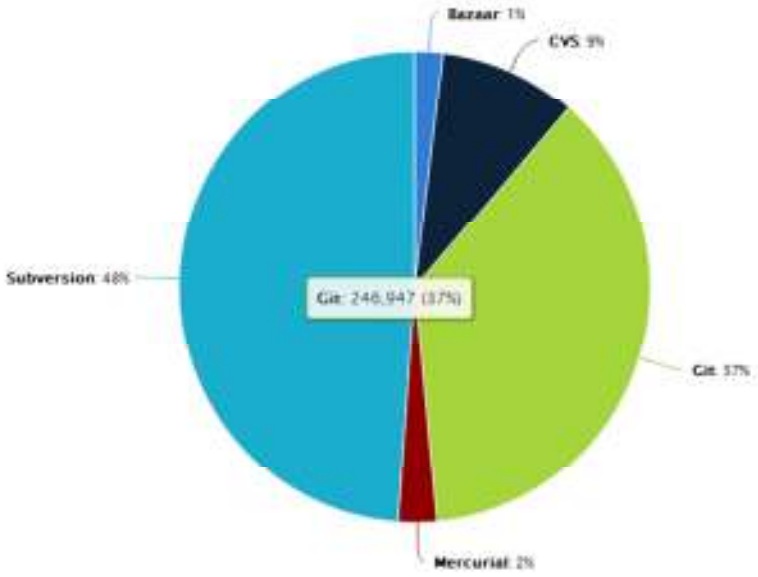
A. Latar Belakang Masalah

Komunitas CAH UNP (KCU) sudah berumur empat tahun sejak komunitas didirikan tahun 2010, akan tetapi belum menunjukkan hasil karya bidang teknologi informasi dan belum pernah memenangkan perlombaan baik tingkat daerah maupun nasional. Didalam proses kegiatannya, mahasiswa yang tergabung dalam komunitas sedikit sekali yang aktif disetiap pertemuan rutin, komunitas juga belum memiliki media untuk belajar bersama, khususnya ketika membuat proyek dan mengembangkan aplikasi yang dikerjakan secara tim, *open recruitment* komunitas dari tahun 2011 sampai 2013 menunjukkan hasil bahwa mahasiswa Fakultas Teknik Universitas Nusantara PGRI Kediri sedikit sekali yang tertarik untuk bergabung di KCU.

Berkomunitas telah dimodelkan oleh Romi Satria Wahono (2007:15) yang mengatakan bahwa model motivasi komunitas cukup efektif digunakan untuk pengembangan dan penerapan sistem pembelajaran elektronik. Kebutuhan akan metode pembelajaran yang lebih efektif dan efisien, pemanfaatan teknologi informasi untuk pendidikan menjadi tidak terelakkan lagi, sehingga perlu dibangun sebuah media pembelajaran untuk menyelesaikan permasalahan tersebut. Merancang *Git Server* merupakan

salah satu solusi untuk menyelesaikan permasalahan, karena *Git Server* berkaitan dengan suatu tim, diskusi, dan kelompok belajar, sehingga *Git Server* dapat diimplementasikan di lingkup pendidikan.

Lima tahun terakhir ini *Git* sangat popular, dipopulerkan melalui *GitHub*. *Git* banyak digunakan oleh para pengembang perangkat lunak karena kemudahan dan kecepatan dalam model percabangan, menciptakan, penggabungan dan bekerja dengan cabang menjadi mudah dan cepat. statistik yang di ambil dari ohloh.net *git* menunjukkan perkembangan sangat cepat :



Gambar 1.1 Perbandingan *Repositories*

"Dalam pengembangan *software*, *Git* adalah sebuah *Distributed Revision Control* (*DRC*) dan *Source Code Management* (*SCM*) *System* dengan penekanan pada kecepatan." (Torvalds, 2005:list.linux-kernel). Jadi *Git Server* dirancang untuk melayani pendistribusian *source code* hasil

belajar mahasiswa, yang mana *Git* yang dioperasikan secara personal dapat terdistribusi dengan adanya *Git Server*.

"*GitHub* adalah situs *repository source code* terbuka yang paling populer" (Finley, 2011:homepage), *Git Server* dirancang dengan pendekatan *GitHub social coding* yaitu pendekatan bagaimana *git server* mensosialisasikan *source code* hasil belajar mahasiswa secara terbuka, sehingga *Git Server* menjadi populer seperti *GitHub* dalam lingkup internal universitas.

"Sejalan dengan pendekatan konstruktivisme dalam pembelajaran, salah satu model pembelajaran yang kini banyak mendapat respon adalah model pembelajaran kooperatif (bekerja bersama - sama)" (Sumarno, 2011:homepage), maka *Git Server* secara tidak langsung dapat digunakan sebagai media pembelajaran kooperatif.

Pada Penelitian ini *Git Server* akan dirancang untuk meningkatkan motivasi belajar bersama mahasiswa yaitu dari media belajar bersama personal menjadi media belajar bersama yang terdistribusi, karena *Git Server* dapat dimanfaatkan seperti, mengerjakan tugas kelompok mata kuliah pemrograman, mengembangkan aplikasi secara bersama – sama, berbagi *source code* dengan cepat dan mudah, serta siapapun dapat melihat dan men-*dowload source code* yang diunggah ke *Git Server*.

Diharapkan dengan adanya *Git Server* dengan cara mensosialisasikan *source code* secara terbuka dapat meningkatan belajar bersama mahasiswa, sehingga didapatkan proses dan hasil yang optimal.

B. Identifikasi Masalah

Berdasarkan latar belakang masalah tersebut maka perlu dibangun sebuah rancangan *Git Server* dengan pendekatan *GitHub social coding* yang dapat digunakan sebagai media belajar bersama mata kuliah pemrograman sehingga dapat membantu meningkatkan pembelajaran mahasiswa.

C. Pembatasan Masalah

Pembatasan masalah yang dikaji dalam skripsi ini adalah:

1. *Git Server* dirancang terfokus pada kegunaannya sebagai media untuk belajar bersama mata kuliah pemrograman.
2. Pendekatan *GitHub social coding* dirumuskan sebagai cara mensosialisasikan hasil belajar bersama.
3. Peningkatan pembelajaran mahasiswa terbatas pada mahasiswa dapat belajar bersama menggunakan *Git Server*.

D. Rumusan Masalah

1. Bagaimana merancang *Git Server* dengan pendekatan *GitHub Social Coding* dalam peningkatkan pembelajaran mahasiswa ?
2. Apakah rancangan *Git Server* dengan pendekatan *GitHub Social Coding* dapat meningkatkan pembelajaran mahasiswa ?

E. Tujuan Penelitian

1. Untuk merancang *Git Server* dengan pendekatan *GitHub Social Coding*

dalam peningkatan pembelajaran mahasiswa.

2. Untuk mengetahui rancangan *Git Server* dengan pendekatan *GitHub Social Coding* dapat meningkatkan pembelajaran mahasiswa pada mata kuliah pemrograman.

F. Kegunaan Penelitian

1. Teoritis

   Kegunaan secara teoritis dapat dikelompokkan menjadi tiga yaitu terhadap peneliti, perguruan tinggi, dan ilmu pengetahuan.

   a. Peneliti

      Merupakan sumbangsih bagi peneliti kepada mahasiswa lain tentang rancangan *Git Server* sehingga dapat dikembangkan untuk penelitian selanjutnya.

   b. Perguruan Tinggi

      Merupakan wujud Tridharma bagi perguruan tinggi yaitu sebagai pelaksana Penelitian.

   c. Ilmu Pengetahuan

      Dapat dijadikan sebagai rujukan penelitian bagi mahasiswa atau orang lain untuk menyusun landasan teori.

2. Praktis

   Kegunaan secara praktis dapat di kelompokkan menjadi tiga yaitu terhadap mahasiswa, dosen dan komunitas.

   a. Mahasiswa

Mahasiswa yang tergabung dalam KCU dapat belajar bersama, mengerjakan tugas pemrograman, membuat proyek yang dikerjakan secara tim, kemudian mahasiswa yang tidak tergabung dalam KCU dapat melihat dan men-*download* hasil belajar bersama secara terbuka kapanpun serta dapat dimanfaatkan sebagai media belajar mata kuliah pemrograman.

b. Dosen

Dosen pemrograman dapat memberikan tugas kelompok mahasiswa dengan menggunkan *Git Server* sebagai pengontrol tugasnya, sehingga mahasiswa yang aktif dan tidak aktif dalam mengerjakan tugasnya dapat di lihat pada *log* sistemnya.

c. Komunitas

Komunitas dapat menerapkan program kerjanya dengan mudah kepada anggotanya seperti mengikuti perlombaan, membuat proyek bidang teknologi informasi, serta secara tidak langsung perannya, dapat bermanfaat bagi banyak mahasiswa.

G. Sistematika Penulisan

Sekripsi ini disusun ke dalam lima bab dimana secara garis besar skripsi ini berisi :

BAB I : PENDAHULUAN

Pada bab ini akan dijelaskan latar belakang penulisan skripsi, identifikasi masalah, pembatasan masalah, rumusan masalah, tujuan masalah, kegunaan,

serta sistematika penulisan skripsi.

BAB II : LANDASAN TEORI

Pada bab ini akan dijelaskan teori – teori mendasar sebagai pendukung dan landasan penulisan skripsi, serta menjelaskan kebutuhan alat dan teknologi untuk merancang sistem *Git Server*.

BAB III : METODE PENGEMBANGAN

Pada bab ini akan diuraikan tentang model pengembangan, prosedur pengembangan, lokasi dan subyek Penelitian, uji coba model, dan teknik analisis data.

BAB IV : DESKRIPSI, INTERPRETASI DAN PEMBAHASAN

pada bab ini akan diberikan hasil studi pendahuluan, pengujian, dan pembahasan hasil Penelitian.

BAB V : SIMPULAN, IMPLIKASI DAN SARAN

pada bab ini akan diberikan simpulan dari seluruh hasil analisis dan perancangan pada Penelitian yang telah dilakukan disertai saran untuk pengembangan lebih lanjut.

BAB II

LANDASAN TEORI

A. Pendahuluan

Selain *Git*, Ada sekitar sepuluh *version control* lain seperti *Bazar*, *Subversion*, *Mercurial*, *CVS*, *RCS*, *Perforce*, *ClearCase*, *Gnu Arch*, *GNU CSSC*, dan *BitKeeper* (Fajar, 2014). *Version control* adalah sebuah sistem yang mencatat setiap perubahan terhadap sebuah berkas atau kumpulan berkas sehingga pada suatu saat anda dapat kembali kepada salah satu versi dari berkas tersebut (Chacon, 2009:1). Jadi, *Git* adalah sekian dari beberapa *version control* yang ada. Dalam bukunya Loeliger (2012:1) mengatakan bahwa sebuah alat yang mengelola dan melacak versi yang berbeda dari perangkat lunak atau konten lainnya yang umum disebut sebagai *version control system* (VCS), *Source Code Management* (SCM), *Revision Control System* (RCS) dan beberapa permutasi lain dari kata-kata 'Revisi', 'versi', 'Kode', 'Isi', 'Kontrol', 'Manajemen', dan 'Sistem'.

Menurut Gadja (2013:1-6) untuk menginstal *Git* dan memulai *Git* adalah seperti berikut:

1. Instalasi *Git*
   a Instalasi *Git* di *Windows*

     Pergi ke alamat http://msysgit.github.io/ dan *download* versi *installer* terbaru untuk *windows*, jalankan *installer* dan abaikan semua

pilihan, atur ke nilai *default*. Setelah ini, *Git* siap dijalankan di sistem anda.

Dimana ada dua metode untuk menginstall *Git* untuk *windows* pertama gunakan paket *cygwin* yang tersedia di http://www.cygwin.com kedua gunakan *installer standalone* menggunakan *msysgit*.

b Instalasi *Git* di *Linux*

Tergantung *system* yang anda jalankan, lakukan suatu perintah:

```
# untuk ubuntu
$ sudo apt-get install git

# untuk fedora
$ sudo yum install git
```

Cara termudah untuk menginstal *Git* di *linux* adalah dengan menggunkan paket yang tersedia, untuk memverifikasi bahwa instalasi berhasil, anda dapat menjalankan perintah:

```
$ git --version
```

Itu akan mencetak versi *Git* yang terinstal pada sistem anda.

2. Memulai *Git*

Untuk mengkonfigurasi *Git* agar dapat bekerja dan berjalan, jalankan perintah berikut:

```
$ git config --global user.name
```

Itu akan mencetak hasil kosong, itu karena tepat setelah `user.name` tidak dikonfigurasi. Untuk mengatur konfigurasi `user.name` gunakan perintah berikut :

```
$ git config --global user.name "John Doe"
```

Ganti John Doe dengan mengetik nama anda atau nama panggilan.

Selanjutnya, jalankan perintah :

```
$ git config --global user.email "john.doe@example.com"
```

Perintah ini akan mengatur *email* anda. Jika anda ingin membuat *commits* dalam repositori *Git*, anda harus mengkonfigurasi dua pengaturan : `user.name` dan `user.email`.

Jika tidak, ketika anda menjalankan perintah `$ git commit`, *Git* akan mencetak peringatan. *String* yang anda gunakan seperti nilai `user.name` dan `user.email` akan disimpan dalam setiap *commit* yang anda buat.

B. *Git Server*

1. Pengertian

Git adalah sistem manajemen versi dan kode program, tidak seperti sistem *SCM* lainnya (seperti *CVS*, *Subversion* dan lain-lain), *Git* didistribusikan dan mendukung percabangan yang rumit, *Git* dirancang untuk proyek–proyek *open source* oleh Linus Torvalds (Brazdil, 2013:5). sedangkan menurut Linus Torvalds (2005:list.linux-kernel) yang menciptakan *Git* itu sendiri mengungkapkan bahwa "*Git* adalah sebuah *Distributed Revision Control* (*DRC*) dan *Source Code Management* (*SCM*) *System* dengan penekanan pada kecepatan". Jadi *Git* adalah

sebuah perangkat lunak yang dapat melakukan manajemen kode program baik secara personal maupun terdistribusi.

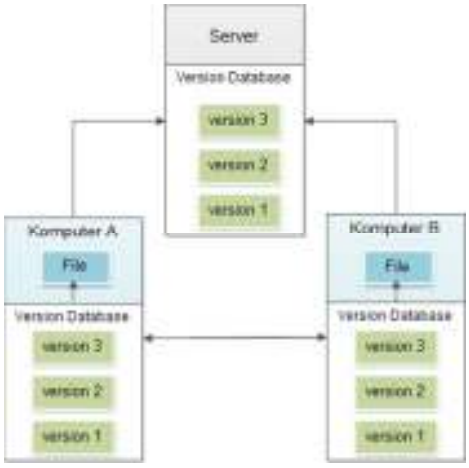
Sedangkan "*Server* adalah sebuah komputer yang menjadi pusat kegiatan suatu jaringan yang dapat memproses satu atau lebih layanan jaringan." (M. Doss, 1999:1), menurut Frans (2002) "*Server* adalah pemilik informasi yang menyediakan dirinya untuk memberikan *service* atau layanan".

Jadi, *Git Server* dapat di artikan sebagai penyedia layanan manajemen kode program yang terdistribusi dalam jaringan.

2. Kegunaan *Git*

Sesuai pengertiannya bahwa *Git* sebagai *Distributed Version Control System (DVCS),* pengertian *Version control* adalah sebuah sistem yang mencatat setiap perubahan terhadap berkas atau kumpulan berkas sehingga pada suatu saat dapat dikembalikan pada salah satu versi dari berkas tersebut.(Chacon, 2009:1). *Git* mampu menangani sejumlah *remote repository* dengan baik, jadi anda dapat melakukan kolaborasi dengan berbagai kelompok kolaborator dalam berbagai cara, secara bersama-sama pada suatu proyek. (Chacon, 2009:3)

Jadi, agar *Git* dapat melakukan *version control* secara terdistribusi dan dapat menangani sejumlah *remote repository,* maka perlu adanya *Git Server* untuk menyediakan layanan tersebut. Seperti diperlihatkan pada gambar berikut :



Gambar 2.1. Diagram *Distributed Version Control* (Chacon, 2009:3)

3. Konfigurasi *Git*

Menurut Chacon (2009:8–9) tahapan–tahapan konfigurasi *Git* adalah sebagai berikut:

a. Instalasi *Git*

1) Instalasi *Git* di *Windows*

Menginstall *Git* pada *windows* sangatlah mudah. Proyek *msysGit* memakai sebuah prosedur instalasi lebih mudah. Cukup *download file installer*-nya dari halaman *Google Code*, dan menjalankannya:

http://code.google.com/p/msysgit

Setelah terinstal, Anda akan memperoleh versi *command-line* (bersama dengan *SSH* klien) dan *GUI* standar.

2) Instalasi *Git* di *Linux*

Jika, menginstall *Git* di *Linux* menggunakan *installer biner*, bisa melakukannya melalui alat manajemen paket yang

sesuai dengan distribusi *linux* yang digunakan. Jika menggunkan *Fedora*, dapat menggunakan perintah `yum`:

```
$ yum install git-core
```

Jika menggunakan distribusi berbasis *debian* seperti *ubuntu*, coba `apt-get`:

```
$ apt-get install git-core
```

b. Konfigurasi Awal *Git*

Hal pertama yang harus lakukan setelah menginstalkan *Git* adalah mengatur *username* dan alamat e-*mail*. Hal ini penting karena setiap *commit* pada *Git* menggunakan informasi ini, dan akan selamanya disimpan dengan *commit* yang dibuat tersebut:

```
$ git config --global user.name "Nama Lengkap"
$ git config --global user.email "user@email.com"
```

c. Menggunakan *Git*

Ketika bekerja dengan *Git Server* sebelumnya harus bisa mengoperasikan *Git* secara personal kemudin setelah mahir dengan perintah dasar berikutnya perintah kolaborasi dengan *Git Server*. Menurut Loeliger (2012:21–23) berikut adalah perintah dasar menggunkan *Git*:

1) Membuat *Repository* Awal

Tipe model yang khas digunakan, mari membuat repositori untuk *website* pribadi dari direktori `~/public_html` dan tempatkan itu sebagai *Git repository*.

Jika tidak memiliki konten *website* pribadi di `~/public_html`, buatlah direktori dan tempatkan beberapa konten sederhana dalam sebuah *file* yang benama `index.html` :

```
$ mkdir ~/public_html
$ cd ~/public_html
$ echo 'My website is alive!' > index.html
```

Untuk mengaktifkan `~/public_html` atau direktori apapun ke *Git repository*, jalankan `git init`:

2) Perintah `$ git init`

Perintah ini menciptakan subdirektori baru bernama `.git` yang berisi semua *file* repositori yang diperlukan – untuk sebuah kerangka *Git repository*. Pada awalnya, tidak ada proyek yang terlacak. (Chacon, 2009:13).

```
$ git init
Initialized empty Git repository in .git/
```

3) Perintah `$ git add <filename>`

Perintah `$ git init` itu membuat *Git repository* baru, awalnya *Git repository* kosong. Untuk mengelola konten, maka secara gamblang harus disimpan ke dalam *repository*.

Gunakan `$ git add` *file* untuk menambahkan *file* ke *repository* :

```
$ git add index.html
```

Jika sebuah direktori diisi dengan beberapa *file*, biarkan *Git* menambahkan semua *file* dalam direktori dan semua subdirektori dengan `$ git add .` (Argumen `.`, titil tunggal atau

"dot" atau dalam bahasa *unix*, adalah istilah untuk direktori saat ini.)

4) Perintah `$ git status`

Setelah `add`. *Git* tahu bahwa *file*, `index.html`, adalah tetap dalam *repository*. Namun sejauh itu, *Git* hanya men-*staged file*, langkah sementara sebelum `commit` (berkomitmen). *Git* memisahkan `add` dan `commit` sebagai langkah untuk menghindari volatilitas. Menjalankan `git status` ini mengungkapkan diantara keadaan `index.html`:

```
$ git status

# On branch master
#
# Initial commit
#
# Changes to be committed:
# (use "git rm --cached <file>..." to unstage)
#
# new file: index.html
```

Laporan perintah *file* baru `index.html` akan ditambahkan ke *repository* setelah berikutnya di *commit*.

5) Perintah `$ git commit –m "pesan perubahan"`

Selain ke perubahan yang sebenarnya ke direktori dan isi *file*, *Git* mencatat beberapa bagian lain dari metadata dengan setiap `commit`, termasuk pesan *log* dan perubahan *author*. yang memenuhi syarat perintah `$ git commit` menyediakan pesan *log* dan *author*:

```
$ git commit -m \
"Initial contents of public_html" \
--author="Jon Loeliger <jdl@example.com>"
```

```
Created initial commit 9da581d: Initial contents of public_html
1 files changed, 1 insertions(+), 0 deletions(-)
create mode 100644 index.html
```

Setelah melakukan penambahan *file* baru ke dalam *repository*, `git status` menunjukkan bahwa tidak ada yang belum diselesaikan, perubahan *staged* yang akan di *commit*.

```
$ git status
# On branch master
nothing to commit (working directory clean)
```

Git juga meluangkan waktu untuk memberitahu bahwa *working direktory* bersih, yang berarti *working direktory* tidak diketahui atau memodifikasi *file* yang berbeda dari yang ada pada *repository*.

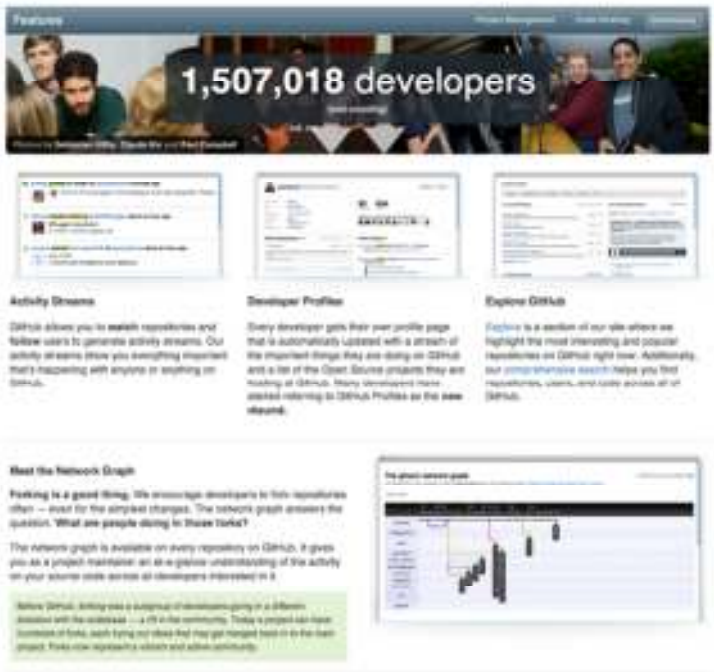
4. *GitHub Social Coding*

a. Pengertian

GitHub adalah layanan untuk *hosting git repository*. *GitHub* memiliki web kode peramban dan memungkinkan mengomentari kode. *GitHub* bukan *open source* tetapi proyek-proyek *open source* dapat di-*hosting* di sana dengan gratis. Untuk proyek-proyek lain, *GitHub* menyediakan *private repository* prabayar dan menawarkan *GitHub Enterprise* untuk instalasi di rumahan (Brazdil, 2013: 10). *GitHub* sangat populer - memiliki lebih dari tiga juta pengguna (Sanheim, 2013:homepage).

GitHub menawarkan aspek-aspek sosial *Web*, seperti yang terlihat di *Twitter*, *Facebook*, dan jejaring sosial lainnya. Dengan konsep *whatching* pengguna lain yang melihat akan membuat tertarik

untuk berkontribusi, *forking repository* untuk mengizinkan siapa pun menyalin sebuah proyek, *pull requests* sebagai sinyal pemilik proyek yang programmer lain memiliki set dari kode yang menarik untuk berpotensi bergabung kedalamnya, dan *line-level comments* pada `commit` untuk mengijinkan perbaikan ulang sederhana untuk berkontribusi, *GitHub* telah membuat aktivitas *social coding* (Loeliger, 2012:390).



Gambar 2.2. *Social Coding* (Loeliger, 2012:391)

Jadi, *Github Social Coding* adalah *hosting git repository* yang memiliki layanan *whatching*, *forking*, *pull requests*, dan *line-level comments*, sehingga memungkinkan pengguna dapat melakukan aktivitas *social coding* secara *open source* maupun *close source*.

b. Langkah–langkah menggunakan *GitHub*

Berkenaan dengan langkah–langkah menggunkanan *Github* Jon Loeliger dan Mattew McCullough (2012:385–398) menjabarkan sebagai berikut :

1) Membuat Akun *GitHub*

Membuat akun dimulai dengan membuka http://github.com di *web browser* dan klik *link sign up* seperti pada gambar:

Gambar 2.3. *GitHub Homepage* (Loeliger, 2012:386)

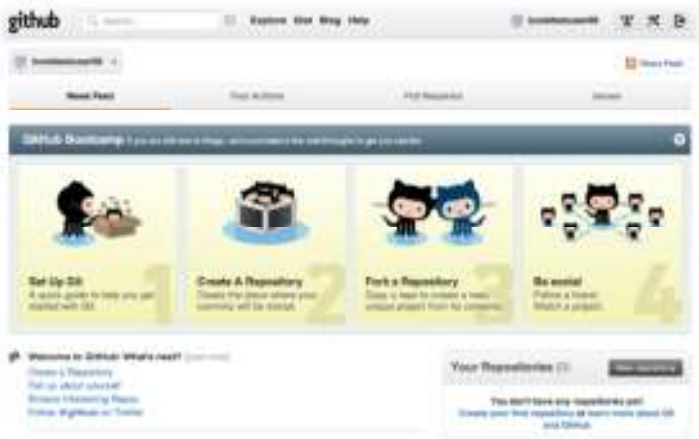
Kemudian anda bisa memilih harga sesuai kebutuhan, jika menggunakan *free account* (akun gratis) maka anda hanya akan mendapatkan satu *private repository* dan yang lainnya yaitu *repository* yang anda ciptakan akan bersifat *public Repository.* Seperti pada gambar berikut :

Gambar 2.4. *Choosing an account type* (Loeliger, 2012:386)

GitHub memiliki empat jenis akun dan rencana kombinasi: personal gratis, personal prabayar, organisasi gratis dan organisasi prabayar, sebuah akun personal merupakan syarat untuk bergabung ke sebuah organisasi, bijaklah dalam memilih *username* karena hanya satu tindakan mengubah *username* diperbolehkan per–akun. Beberapa alamat *email* dapat dikaitkan untuk satu *username* dan untuk *email* dapat diubah sewaktu-waktu. Dengan demikian, *username* adalah bagian yang paling permanen dari informasi *sign up*.

Gambar 2.5. ***Free Personal Account*** **(Loeliger, 2012:387)**

Kesimpulan pada gambar 2.5. yaitu membuat akun personal gratis. Yang merupakan jenis paling umum, pengguna diarahkan ke halaman bantuan *GitHub*, yang menawarkan tutorial tentang pengaturan parameter konfigurasi yang dibutuhkan pengembang menginstal *Git*.



Gambar 2.6. ***Account Creation Complete*** **(Loeliger, 2012:388)**

2) Membuat *Repository GitHub*

a. Informasi *New Repository*

Setelah membuat akun, membuat repositori semudah mengeklik tombol *New Repository* pada *toolbar* paling atas, terlihat disepanjang waktu setelah *login*, atau dengan menavigasi langsung ke halaman *repository* baru dengan mengetikkan http://github.com/new.

Satu–satunya data yang dibutuhkan adalah nama dari *repository*, sedangkan deskripsi tujuan proyek itu optional dan *URL* halaman rumah repositori menjadi sinyal perhatian dari pengelola (gambar 2.7).

Berikutnya, repositori harus diberikan isi (konten) awal, ada dua pendekatan yang berbeda berdasarkan pada apakah ada atau tidak memiliki *commit* untuk melestarikan.

b. *README Seeding* (Pilihan 1)

Jika langkah pertama untuk berkerja dengan proyek ini adalah untuk membuat *repository GitHub* sebelum menulis kode apapun, anda akan membuat *placeholder* sebagai *commit* awal. Selama *repository* baru dibuat di situs *GitHub*, anda akan disajikan pilihan optional *repository* dengan *file README* awal dan *file* `.gitignore`. proyek ini mneggunakan *file README teks* untuk menjelaskan maksud proyek.

Proyek ini kemudian siap untuk di-*cloning* (penggandaan) dengan perintah `git clone url`, setelah itu kode baru dapat dimulai secara lokal di-`add` dan di-`commit`.

c. *Adding a Remote* (Pilihan 2)

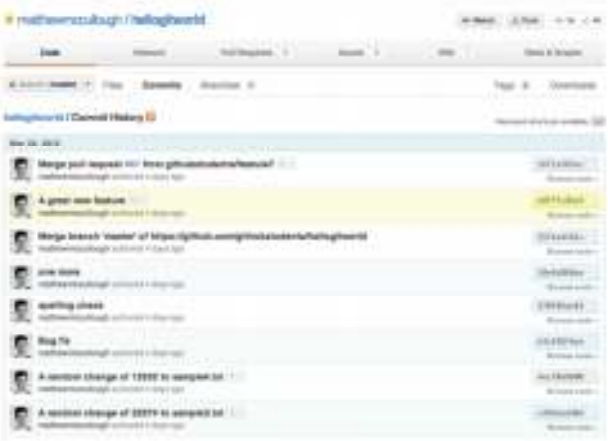
Jika anda telah memiliki repositori *Git* lokal dengan `commit`, anda dapat menghubungkan alamat *GitHub* ke *repo* lokal anda. Caranya dengan menambahkan *GitHub URL* (*git remote*) ke *repository Git* lokal yang ada dengan perintah `git remote add url`.

d. Pushing the Local Content to GitHub

Salah satu dari dua pilihan telah diikuti untuk menghubungkan *repository* lokal ke *remote repository*, isi dari repo lokal dapat di-`push` ke *GitHub*. Hal ini diselesaikan dengan perintah `git push remote branch`, jika branch belum pernah di publikasikan sebelumnya. Ungkapan yang lebih spesifik `git push –u rigin master` yang sesuai, dimana `–u` untuk memberitahu *Git* untuk melacak *push branch*, *push* ke *remote* `origin`, dan hanya untuk *push* ke *branch* `master`.

Gambar 2.7. *Creating a Public Repo* (Loeliger, 2012:389)

Setelah koneksi ke *upstream* (*GitHub*) server didirikan dengan satu teknik sebelumnya, perubahan kode selanjutnya dapat dengan mudah didorong dengan tambahan panggilan `git push`. Hal ini menunjukkan manfaat inti dari akses terpusat *Git* repository, bahkan sangat terdistribusi pekerja yang terfokus dengan alat seperti *Git*: kemampuan untuk melihat perbuhan semua anggota dari proyek ketika telah selesai dan di-*push* (Gambar 2.8), bahkan meskipun mereka sedang *offline*.



Gambar 2.8. *Commit History on GitHub* (Loeliger, 2012:390)

3) *Watchers*

Yang paling sederhana dari fitur *social coding* dapat ditemukan di *GitHub* adalah *Watching*, dimulai dengan menekan tombol *Watch* seperti yang ditunjukkan pada (Gambar 2.9). *Watching*, sebuah konsep serupa dengan pengikut (*Follow*) *Twitter* atau teman (*Friends*) *Facebook*, yaitu sebagai sinyal minat pengguna *GitHub*, organisasi, atau proyek tertentu.

Gambar 2.9. *Watch Button*. (Loeliger, 2012:391)

4) *News Feed*

Selain lima teknik diatas yaitu menonton pengguna, organisasi, atau menyediakan *repository*, itu juga membentuk isi dari *news feed* pribadi seperti yang ditunjukkan pada (Gambar 2.10). *News feed* ini melaporkan kegiatan yang berhubungan dengan *repository* pengguna dan organisasi yang di tonton (*Watching*).

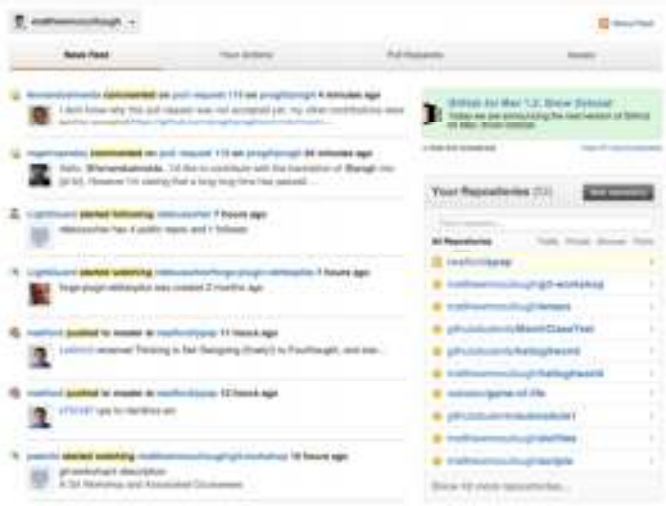
News feed ditawarkan sebagai halaman web dapat dilihat pada situs github.com, serta *RSS feed* untuk konsumsi dalam pembaca aplikasi yang dipilih.

5) *Forks*

Ide berikutnya itu *GitHub* dipopulerkan, begitu banyaknya

sambutan sehingga telah menyebar ke domain lainnya, adalah *fork* personal proyek (Gambar 2.11). *forking* sering diartikan sebuah cara perpisahan agresif dengan cara meng-*copy* dari proyek utama dengan maksud mengambil program dalam arah yang berbeda.

Ide *GitHub* tentang *forking* adalah satu langkah positif yang memungkinkan lebih banyak kontributor untuk membuat lebih banyak yang berkontribusi dengan cara terkontrol dan dapat dilihat. *Forking* adalah kemampuan demokratis dari setiap kontributor yang potensial untuk mendapatkan salinan pribadi dari kode proyek. Salinan pribadi ini (*fork* dalam bahasa *GitHub*) kemudian dapat diubah sesuai keinginan tanpa izin eksplisit dari penulis asli. Hal ini tidak menimbulkan resiko apapun untuk proyek inti karena perubahan yang terjadi dalam *forked repository* (repositori *copy*-an/cabang), bukan *repository* asli.



Gambar 2.10. *News Feed* (Loeliger, 2012:393)

Gambar 2.11. *Fork Button* (Loeliger, 2012:393)

Ini adalah kebalikan dari konsep inti dilindunginya oleh *apache* atau *eclipse project*, dimana *patch* disampaikan sebagai *file* lampiran laporan *bug*. Manfaat utama model ini adalah transparasi dan visibilitas dari sumbangan komunitas (Gambar 2.11), sebelum diserahkan kembali ke proyek inti untuk didiskusikan dan memungkinkan untuk penggabungan.

Network Graph, yang ditunjukkan pada Gambar 2.12, menampilkan hubungan dari proyek inti adalah *branch* dan *commit* untuk orang – orang dari *branch* lain dan *commit* lain, termasuk *forks repository*. Ini memberikan gambaran aktivitas komunitas tingkat tinggi pada proyek dan apakah *fork* yng diberikan menyimpang secara signifikan dari proyek inti.

Gambar 2.12. *Network graph* (Loeliger 2012:394)

6) Membuat *Pull Requests*

Forking adalah langkah yang memungkinkan untuk menciptkan salinan pribadi dari proyek, tetapi nilai nyata proyek inti terletak pada aksi kedua, secara resmi disebut *pull requests*, *pull requests* memungkinkan setiap pengguna dengan *commit*-nya bahwa dia merasa membuat kontribusi yang berguna untuk mengumumkan bahwa ikut berkontribusi kepada pemilik proyek inti.

Setelah kontributor selesai dengan fitur *coding*, melakukan *commit* kode untuk nama *branch* baru, dan *push branch* baru untuk *fork*, itu dapat berubah dengan *pull requests*. *Pull requests* bisa secara akurat, apabila minimalnya menggambarkan daftar *commit* topiknya terfokus. *Pull requests* paling sering didasarkan pada seluruh isi cabang topik.

Gambar 2.13. *Pull Requsts Button* (Loeliger, 2012:395)

5. Alat *Git Server*

a. Kebutuhan *Hardware*

Persyaratan Sistem

Ubuntu 12.10 Server Edition mendukung tiga arsitektur besar: *Intel x86, AMD64,* dan *ARM*. Tabel di bawah ini adalah daftar spesifikasi *hardware*.

Tabel 2.1. Persyaratan Minimum yang direkomendasikan.

Tipe Instalasi	CPU	RAM	Hard Drive Space	
			Base System	All Task Installed
Server (Standar)	1 GHz	512 MB	1 GB	1.75 GB
Server (Minimal)	300 MHz	256 MB	700 MB	1.4 GB

Edisi *Server* menyediakan dasar umum untuk segala macam aplikasi *server*. Ini adalah desain minimalis menyediakan *platform* untuk layanan yang diinginkan, seperti layanan *file* / *print*, *web hosting*, *email hosting*, dll. (Williamson, 2012:4)

b. Kebutuhan *Software*

Tabel 2.2. *Packet Installed* (Loeliger, 2012:10)

Sistem Operasi	Packet Installed
Linux Ubuntu Server 12.04.1 LTS	Git, gitweb, git-daemon-run, gitolite, openssh-server, bind9, isc-dhcp-server, apache2.

6. Langkah–langkah Merancang *Git Server*

Menjalankan *Git server* itu sederhana. yaitu, memilih protokol yang ingin server berkomunikasi dengannya. Sebuah *remote repository* secara umum adalah merupakan sebuah *bare repository*, sebuah *Git repository* yang tidak memiliki *working direktory*, karena *repository* hanya digunakan sebagai titik kolaborasi. Dalam istilah sederhana *bare repository* adalah isi dari direktori `.git` proyek anda dan tidak ada yang lain. Langkah–langkah merancang *Git Server,* Scott Chacon (2009:71–88) telah merumuskan sebagai berikut:

a. Protokol

Git mnggunkan empat protokol jaringan untuk mentrasfer data, lokal, *Secure Shell* (*SSH*), *Git*, dan *HTTP*.

1) Lokal Protokol

Yang paling mendasar adalah protokol lokal, yang mana me-*remote reposiroy* di direktori lain di disk. Hal ini sering digunakan jika semua orang dalam tim memiliki akses ke file system bersama seperti *NFS* (*Network file System*), atau dalam

beberapa kasus mungkin bahwa setiap orang *login* ke dalam komputer yang sama.

Jika anda memiliki *file system* yang di-*mount* bersama, maka anda bisa *clone*, *push*, dan *pull* repositori berbasis lokal. Untuk meng-*cloning* repositori seperti ini atau menambahkan *remote* proyek yang sudah ada, gunakan *path repository* sebagai *URL*. Misalnya, untuk meng-*cloning* repositori lokal, anda bisa menjalankan sesuatu seperti ini:

```
$ git clone /opt/git/project.git
```

Atau anda bisa melakukan ini:

```
$ git clone file:///opt/git/project.git
```

Untuk menambakan *repository* lokal proyek *Git* yang ada, anda dapat menjalankan sesuatu seperti ini:

```
$ git remote add local_proj \
/opt/git/project.git
```

Kemudian anda bisa *push* dan *pull* dari jauh yang seolah-olah anda melakukan dalam jaringan.

2) Protokol *SSH*

Mungkin protokol *transport* yang paling umum untuk *Git* adalah *SSH*. Hal ini karena akses *SSH* ke server sudah diatur di sebagian tempat. *SSH* juga satu-satunya protokol berbasis jaringan yang anda dapat dengan mudah membaca (*read*) dan menulis (*write*) proyek. Dua protokol jaringan lainnya (*HTTP* dan

Git) umumnya hanya *read-only*, anda masih perlu *SSH* untuk menulis perintah anda sendiri.

Untuk meng-*cloning* repositori *Git* melalui *SSH*, anda dapat menggunakan *SSH*://*URL* seperti ini:

```
$ git clone ssh://user@server:project.git
```

Atau anda bisa tidak menentukan protokol – *Git* sudah mengasumsikan *SSH* jika anda tidak menentukan protokol :

```
$ git clone user@server:project.git
```

3) Protokol Git

Berikutnya adalah protokol *Git*, ini adalah *daemon* khusus yang dikemas dengan *Git*, protokol ini *listen port* khusus (9418) yang menyediakan layanan yang mirip dengan protokol *SSH*, tetapi sama sekali tidak ada otentikasi. Agar repositori melayani protokol *Git*, anda harus membuat *file* `git-expor-daemon-ok`, *daemon* tidak akan melayani tanpa *file* tersebut didalamnya, tapi selain itu tidak ada keamanan. Sehingga repositori *Git* tersedia bagi semua orang untuk meng-*cloning* maupun tidak, ini berarti pada umumnya tidak ada *push* melalui protokol ini.

4) Protokol *HTTP/S*

Keindahan protocol *HTTP* atau *HTTPS* adalah kemudahan dalam pengataurannya. Pada dasarnya, yang harus anda lakukan adalah meletakkan *bare repository Git* dibawah *document root HTTP* anda dan mengatur secara khusus `post-update` *hook*, siapapun dapat mengakses *web server* yang anda meletakkan

repositori juga dapat meng-*cloning* repositori anda. Untuk memberikan akses *read* ke repositori anda melalui *HTTP*, lakukan sesuatu seperti ini :

```
$ cd /var/www/htdocs/
$ Git clone --bare /patch/to/git_repository
  gitproject.git
$ cd gitproject.git
$ mv hooks/post-update.sample hooks/post-update
$ chmod a+x hooks/post-update
```

`Post-update` *hook* yang datang bersama *Git* secara *default* menjalanakan perintah yang sesuai (`git update-server-info`) untuk membuat *HTTP fetching* dan *cloning* bekerja dengan benar. Perintah ini akan berjalan ketika anda *push* ke repositori ini melalui *SSH*, kemudian orang lain bias *cloning* seperti berikut:

```
$ git clone http://example.com/gitproject.git
```

Dalam kasus ini, kita menggunakan `/var/www/htdocs` jalur yang umum dalam pembuatan *apache*, tetapi anda dapat menggunkan *server web* statis, hanya menempatkan *bare repository* di tempatnya.

Ini memungkinkan untuk membuat *Git push* diatas *HTTP*, meskipun teknik yang tidak banyak digunakan dan mengharuskan anda untuk mengatur persyaratan *WebDAV* yang rumit, karena itu jarang digunakan, jika anda tertarik menggunkan protocol *HTTP-push*, anda dapat membacanya untuk mempersiapkan repositori untuk tujuan ini di:

http://www.kernel.org/pub/software/scm/git/docs/howto/setup-git-server-over-http.txt.

Satu hal yang menyenangkan tentang membuat *Git push* diatas *HTTP* adalah bahwa anda dapat menggunkan *server WebDAV*, tanpa kerumitan fitur *Git*, demikian juga, anda dapat menggunakan fungsi ini jika penyedia *hosting web* anda mendukung *WebDAV* untuk menulis update ke situs *web* anda.

b. Mendapatkan *Git* di *Server*

Pada awalnya untuk menyiapkan *Git Server*, Anda harus mengekspor repositori yang sudah ada menjadi *bare repository* baru, yaitu repositori yang tidak mengandung *working directory*. Ini umumnya mudah dilakukan. Dalam rangka untuk meng-cloning repositori anda untuk membuat *bare repository* baru, anda dapat menjalankan perintah *clone* dengan opsi --*bare*. Dengan konvensi, direktori *bare repository* berakiran `.git`, seperti :

```
$ git clone --bare my_project my_project.git
Initialized empty Git repository in
/opt/projects/my_project.git/
```

Keluaran untuk perintah ini mungkin sedikit membingungkan karena `clone` pada dasarnya adalah sebuah `git init` kemudian `git fetch`, kita melihat beberapa keluaran dari bagian `git init`, yang minciptakan direktori kosong. Sebenarnya transfer objek tidak memberikan keluaran, karena itu tidak terjadi. Anda sekarang harus memiliki salinan data direktori *Git* dalam `my_project.git` di

direktori anda. Ini kira- kira setara dengan sesuatu seperti ini:

```
$ cp –Rf my_project/.git my_project.git
```

Beberapa perbedaan kecil pada *file* konfigurasi , tapi untuk tujuan anda, ini sudah mendekati sama. Dibutuhkan hanya *Git repository* saja, tanpa *working directory*, dan menciptakan direktori khusus itu saja.

1) Meletakkan *Bare Repository* pada *Server*

Sekarang anda memiliki salinan *bare repository*, semua yang perlu anda lakukan adalah meletakkannya di *server* dan mengatur protokol anda. Katakanlah anda telah menyiapkan *server* yaitu `git.example.com` dan anda memiliki akses *SSH* kesana, kemudian anda ingin menyimpan semua repositori *Git* anda dibawah direktori `/opt/git/` . anda dapat mengatur repositori baru anda dengan menyalin *bare repositori* diatas:

```
$ scp –r my_project.git \
  user@git.example.com:/opt/git
```

Pada bagian ini, pengguna lain yang memiliki akses *SSH* ke *server* yang sama, yang telah mendapatkan akses *read* ke direktori `/opt/git` dapat meng-*cloning* repositori anda dengan menjalankan:

```
$ git clone \
user@git.example.com:/opt/git/my_project.git
```

Jika pengguna *SSH* masuk kedalam sebuah *server* dan memiliki akses tulis ke direktori `/opt/git/my_project.git`, mereka juga akan secara otomatis memiliki akses *push*. *Git*

secara otomatis akan menambah akses *group write* ke reposotri dengan baik jika anda menjalankan perintah `git init` dengan opsi `--shared`.

```
$ ssh user@git.example.com
$ cd /opt/git/my_project.git
$ git init --bare --shared
```

Anda lihat betapa mudahnya untuk mengambil repositori *Git*, mmbuat versi *bare*, dan meletakkan di-*server* yang anda dan kolaborator memiliki akses *SSH*. Sekarang anda siap berkolaborasi pada proyek yang sama.

2) *Small Setups*

Salah satu aspek yang paling rumit menyiapkan Git *server* adalah menejemen *user*. Jika anda menginginkan beberapa repositori untuk *read-only* bagi pengguna tertentu dan *read*/*write* ke orang lain, akses dan hak akses akan sedikit sulit untuk mengaturnya.

3) Akses *SSH*

Jika anda sudah memiliki sebuah *server* yang semua pengembang anda memiliki akses *SSH*, itu umumnya paling mudah untuk mengatur repositori pertama anda disana, karena yang anda lakukan hampir tidak ada. Jika anda ingin lebih kompleks mengontrol akses dan hak akses repositori anda, anda dapat menangani mereka dengan hak akses *file* sistem normal dari sistem operasi *server* anda yang berjalan.

Jika anda ingin menempatkan repositori anda pada *server* yang tidak memiliki akun untuk semua orang di tim anda yang semua ingin memiliki akses *write*, maka anda harus mengatur akses *SSH* untuk mereka, kami berasumsi bahwa jika anda memiliki sebuah *server* yang dapat digunakan untuk melakukan hal ini, maka anda sudah memiliki *server SSH* terinstal.

Ada beberapa cara anda dapat memberikan akses kepada semua orang di tim anda, yang pertama mengatur akun semua orang, sederhana tetapi bisa menjadi semakin rumit. Anda mungkin tidak ingin menjalankan `adduser` dan mengatur *password* sementara untuk setiap pengguna.

Metode kedua adalah menciptakan sebuah *single user* 'git' pada mesin, kemudian meminta setiap pengguna yang mendapatkan akses *write* untuk megirimkan kunci publik *SSH*, dan menambahkan kunci tersebut ke file `/.ssh/authorized_keys` anda ke user 'git'. Pada saat itu, setiap orang dapat mengakses mesin melalui user 'git'.

c. Menghasilkan *SSH Public Key*

Ada yang mengatakan, banyak server *Git* mengotentifikasi penggunaan *SSH Public Key*, dalam rangka memberikan *public key*, setiap pengguna harus menghasilkan satu *public key*, jika mereka tidak memiliki. Proses ini mirip di semua sistem operasi. Pertama, anda harus memeriksa untuk memastikan belum memiliki *key*. Secara

default, *key SSH* pengguna disimpan dalam `/.ssh` didirektori pengguna. Anda dapat dengan mudah memeriksa untuk melihat apakah anda sudah memiliki *key* dengan masuk ke direktori tersebut dan melihat isinya.

```
$ cd ~/.ssh
$ ls
authorized_keys2    id_rsa              known_hosts
config              id_rsa.pub
```

anda sedang melihat sepasang *file* benama sesuatu dan sesuatu.pub, dimana sesuatu itu biasanya `id_dsa.pub` dan `id_rsa.pub` *file* `.pub` itu *public key* dan *file* yang lainnya adalah *private key* anda. Jika anda tidak memiliki *file-file* ini (atau bahkan anda tidak memiliki direktori `.ssh`). anda dapat membuat *key* dengan menjalankan sebuah program yang disebut `ssh-keygen`, yang disediakan dengan paket *SSH* pada sistem *Linux* dan *Mac* dan dilengkapi dengan paket *msysgit* pada *windows*.

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key
(/Users/schacon/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in
/Users/schacon/.ssh/id_rsa.
Your public key has been saved in
/Users/schacon/.ssh/id_rsa.pub.
The key fingerprint is:
43:c5:5b:5f:b1:f1:50:43:ad:20:a6:92:6a:1f:9a:3asc
hacon@agadorlaptop.local
```

Pertama menegaskan diamana anda ingin menyimpan key (`.ssh/id_rsa`), dan kemudian meminta dua kali untuk *passphrase*, anda dapat mengkosongkan jika anda tidak ingin mengetik *password* ketika anda menggunakan *key*.

Sekarang setiap pengguna yang melakukan ini harus mengirim *public key* mereka kepada anda atau siapapun yang mengadministrasi *Git* (dengan asumsi anda menggunakan setup *SSH server* yang membutuhkan *public key*). Yang harus mereka lakukan adalah menyalin isi dari *file* `.pub` dan mengirim ke *email*. *Public key* terlihat seperti ini:

```
$ cat ~/.ssh/id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAklOUpkDHrfHY17SbrmTIp
NLTGK9Tjom/BWDSUGPl+nafzlHDTYW7hdI4yZ5ew18JH4JW9j
bhUFrviQzM7xlELEVf4h9lFX5QVkbPppSwg0cda3Pbv7kOdJ/
MTyBlWXFCR+HAo3FXRitBqxiX1nKhXpHAZsMciLq8V6RjsNAQ
wdsdMFvSlVK/7XAt3FaoJoAsncM1Q9x5+3V0Ww68/eIFmb1zu
UFljQJKprrX88XypNDvjYby6vw/Pb0rwert/EnmZ+AW4OZPnT
PI89ZPmVMLuayrD2cE86Z/il8+gw3r3+1nKatmIkjn2so1d01
QraTlMqVSsbxNrRFi9wrf+M7Q==
schacon@agadorlaptop.local
```

d. Menyiapkan *Server*

Mari kita berjalan melalui pengaturan *SSH* dari sisi *server*. Dalam hal ini anda akan menggunakan metode `authorized_keys` untuk otentikasi pengguna anda. Pertama, anda membuat user 'git' dan sebuah direktori `.ssh` untuk user tersebut.

```
$ sudo adduser git
$ su git
$ cd
$ mkdir .ssh
```

Selanjutnya, anda perlu menambahkan beberapa *public key* pengembang ke *file* `authorized_key` untuk pengguna tersebut. Mari kita asumsikan anda telah menerima beberapa *key* melalui *email* dan

menyimpannya ke temporary *files*. Sekali lagi kunci publik terlihat seperti :

```
$ cat /tmp/id_rsa.john.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCB007n/ww+ouN4gSLKs
sMxXnBOvf9LGt4LojG6rs6hPB09j9R/T17/x4lhJA0F3FR1rP
6kYBRsWj2aThGw6HXLm9/5zytK6Ztg3RPKK+4kYjh6541NYsn
EAZuXz0jTTyAUfrtU3Z5E003C4oxOj6H0rfIF1kKI9MAQLMdp
GW1GYEIgS9EzSdfd8AcCIicTDWbqLAcU4UpkaX8KyGlLwsNuu
GztobF8m72ALC/nLF6JLtPofwFBlgc+myivO7TCUSBdLQlgMV
OFq1I2uPWQOkOWQAHukEOmfjy2jctxSDBQ220ymjaNsHT4kgt
Zg2AYYgPqdAv8JggJICUvax2T9va5 gsg-keypair
```

Anda tinggal menambahkan mereka ke *file* `authorized_key` anda:

```
$ cat /tmp/id_rsa.john.pub >> ~/.ssh/authorized_keys
$ cat /tmp/id_rsa.josie.pub >> ~/.ssh/authorized_keys
$ cat /tmp/id_rsa.jessica.pub >> ~/.ssh/authorized_keys
```

Sekarang, anda dapat mengatur repositori kosong bagi mereka dengan menjalankan `git init` dengan opsi `--bare`, yang menginisialisasi repositori tanpa direktori kerja.

```
$ cd /opt/git
$ mkdir project.git
$ cd project.git
$ git init --bare
```

Kemudian, John, Joise, atau Jesica *push* versi pertama dari proyek mereka ke dalam repositori yang dengan menambahkannya sebagai *remote* dan *push* sebuah *branch*, perhatikan bahwa seseorang harus *shell* ke mesin dan membuat *bare repositori* setiap kali anda ingin menambhakan sebuah proyek. Mari kita gunakan `gitserver` sebagai nama *host* dari *server* dimana anda telah menyiapkan user 'git' dan repositori. Jika anda menjalankan secara internal, dan anda

mengatur *DNS* untuk `gitserver` untuk merujuk ke *server* itu, maka anda menggunakan perintah cukup banyak seperti:

```
# on Johns computer
$ cd myproject
$ git init
$ git add .
$ git commit –m "initial commit"
$ git remote add origin \
  git@gitserver:/opt/git/project.git
$ git push origin master
```

Pada tahapan ini, yang lain bisa meng-*cloning* dan *push* perubahan itu kembali dengan mudah:

```
$ git clone git@gitserver:/opt/git/project.git
$ vim README
$ git commit –am "fix for the README file"
$ git push origin master
```

Dengan metode ini, anda dapat dengan cepat mendapatkan akses *read/write Git server* dan berjalan bersama beberapa pengembang.

Sebagai tindakan pencegahan ekstra, anda dapat dengan mudah membatasi pengguna 'git' untuk hanya melakukan kegiatan Git dengan alat *shell* terbatas yang disebut `git-shell` yang bersama dengan *Git*. Jika anda mendapatkan ini sebagai user 'git' *shell login* anda, maka user 'git' tidak dapat memiliki akses *shell* yang normal ke *server* anda. Untuk menggunakan ini, tentukan `git-shell` bukan *bash* atau *csh* untuk *shell login* pengguna anda. Untuk melakukannya, anda mungkin harus mengedit *file* `/etc/password` anda:

```
$ sudo vim /etc/passwd
```

Pada bagian bawah anda harus menemukan garis yang terlihat seperti ini:

```
git:x:1000:1000::/home/git:/bin/sh
```

Ubah `/bin/sh` ke `/usr/bin/git-shell` (atau *run* `git-shell` untuk melihat dimana itu diinstal). Jalur ini harus terlihat seperti ini:

```
git:x:1000:1000::/home/git:/usr/bin/git-shell
```

Sekarang pengguna 'git' hanya dapat menggunakan koneksi *SSH* untuk *push* dan *pull* repositori Git dan tidak bisa *shell* ke mesin. Jika anda mencoba, akan melihat penolakan login seperti ini:

```
$ ssh git@gitserver
fatal: What do you think I am? A shell?
Connection to gitserver closed.
```

e. *Public Access*

Bagaimana jika anda ingin *anonim* mengakses *read* untuk proyek anda? mungkin bukan *hosting* proyek pribadi internal, anda ingin meng-*host* sebuah proyek *open source*. Atau mungkin anda memiliki sekelompok otomatis membangun *server* atau *server* integrasi berkesinambungan yang berbuah banyak, dan anda tidak ingin harus menghasilkan *SSH key* sepanjang waktu, anda hanya ingin menambahkan akses *read anonim* sederhana.

Mungkin cara paling sederhana untuk *setup* yang lebih kecil adalah dengan menjalankan *web server* statis dengan *document root* dimana repositori *Git* anda, dan kemudian mengaktifkan *hook* `post update`. Katakanlah anda memiliki repositori dalam direktori `/opt/git`, dan *server apache* yang berjalan pada mesin anda. sekali lagi, anda dapat menggunakan *web server* untuk ini, tetapi sebagai

contoh, kita akan menunjukkan beberapa konfigurasi dasar *apache* yang seharusnya memberi anda gambaran tentang apa yang mungkin anda butuhkan.

Pertama, anda harus mengaktifkan *hook*:

```
$ cd project.git
$ mv hooks/post-update.sample hooks/post-update
$ chmod a+x hooks/post-update
```

Jika anda menggunakan versi *Git* lebih awal dari 1.6 perintah `mv` tidak diperlukan – *Git* mulai dengan penamaan *hooks* dengan *postfix* `.sample` baru – baru ini. Apa yang terjadi setelah `post-update` dilakukan? Pada dasarnya terlihat seperti ini:

```
$ cat .git/hooks/post-update
#!/bin/sh
exec git-update-server-info
```

Ini berarti ketika anda *push* ke *server* melalui *SSH*, *Git* akan menjalankan perintah ini untuk memperbaharui *file* yang diperlukan untuk *fetching HTTP*.

Sealanjutnya anda perlu menambahkan entri *VirtualHost* untuk konfigurasi *Apache* dengan *document root* sebagai direktori *root* proyek *Git* anda. Di sini, kita mengasumsikan bahwa anda memiliki pengaturan *wildcard DNS* untuk mengirim `*.gitserver` seperti:

```
<VirtualHost *:80>
  ServerName git.gitserver
  DocumentRoot /opt/git
  <Directory /opt/git/>
        Order Allow, Dany
        Allow from all
```

```
  </Directory>
</VirtualHost>
```

Anda juga harus mengatus *user group unix* dari direktori `/opt/git` ke `www-data` sehingga *server web* anda dapat *read-access* repositori, karena *Apache* menjalan skrip *CGI* akan (secara *default*) berjalan sebagai pengguna:

```
$ chgrp –R www-data /opt/git
```

Ketika anda me-*restart Apache*, anda harus dapat meng-*cloning* repositori anda di bawah direktori tersebut dengan *URL* proyek anda:

```
$ git clone http://git.gitserver/project.git
```

Dengan cara ini, anda dapat mengatur akses *read* berbasis *HTTP* ke salah satu proyek untuk cukup banyak pengguna dalam beberapa menit.

f. *GitWeb*

Sekarang anda memiliki pengetahuan dasar tentang akses *read*/*write* dan *read-only* ke proyek anda, mungkin anda menginginkan sebuah visualisator berbasis *web* sederhana. *Git* dilengkapi dengan skrip *CGI* yang disebut *GitWeb* yang umum digunakan untuk ini. Anda dapat melihat *GitWeb* digunakan di situs-situs seperti `http://git.kernel.org` (lihat Gambar 2.14).

Gambar 2.14. *GitWeb web-based user interface* (Chacon, 2009:81)

Jika anda ingin memeriksa apakah *GitWeb* terlihat seperti proyek anda, *Git* datang dengan perintah untuk menjalankan sementara jika anda memiliki sbuah server ringan pada sistem anda sperti `lighthhpd` atau `webrick`. Pada mesin *linux*, `lighttpd` sering terinstall, sehingga anda dapat mnjalankannya dengan mengetikkan `git instaweb` dalam direktori proyek anda. jika anda menjalankan *Mac*, *Leopard* datang *preinstalled* dengan *Ruby*, sehingga `webrick` mungkin yang terbaik. Untuk memulai `instaweb` dengan *non-lighttpd*, anda dapat menjalankannya dengan opsi `--httpd`.

```
$ git instaweb --httpd=webrick
[2009-02-21 10:02:21] INFO WEBrick 1.3.1
[2009-02-21 10:02:21] INFO ruby 1.8.6 (2008-03-03)
[universal-darwin9.0]
```

Itu dijalankan pada *server HTTPD* port 1234 dan kemudian secara otomatis *browser web* mulai terbuka pada halaman tersebut.

Ini cukup mudah bagi anda. ketika anda sudah selesai dan ingin menutup *server*, anda dapat menjalankan perintah yang sama dengan opsi `--stop`:

```
$ git instaweb --httpd=webrick --stop
```

Jika anda ingin menjalankan antarmuka *web* pada *server* sepanjang waktu untuk tim anda atau untuk sebuah proyek *open source hosting*, anda harus menyiapkan skrip *CGI* untuk dilayani oleh *web server* normal anda. beberapa distribusi *linux* memiliki paket *gitweb* yang memungkinkan anda dapat menginstal melalui `apt` atau `yum`, jadi mungkin anda ingin mencoba cara yang pertama. Kami akan berjalan melalui menginstall *GitWeb* manual sangat cepat. Pertama, anda perlu untuk mendapatkan kode sumber *Git*, yang *GitWeb* bersama dengannya, dan menghasilkan skrip *CGI* manual.

```
$ git clone \
git://git.kernel.org/pub/scm/git/git.git
$ cd git/
$ make GITWEB_PROJECTROOT="/opt/git" \
   prefix=/usr gitweb/gitweb.cgi
$ sudo cp -Rf gitweb /var/www/
```

Perhatikan bahwa anda harus memberitahu perintah mana untuk menemukan repositori *Git* anda dengan variabel `GITWEB_PROJECTROOT`. Sekarang, anda perlu membuat *Apache* menggunakan *CGI* skrip, yang dapat ditambahka ke *VirtualHost*:

```
<VirtualHost: *:80>
    ServerName gitserver
    DocumentRoot /var/www/gitweb
    <Directory /var/www/gitweb>
      Options ExecCGI +FollowSymLinks \
+SymLinksIfOwnerMatch
        AllowOverride All
        order allow,deny
        Allow from all
```

```
        AddHandler cgi-script cgi
        DirectoryIndex gitweb.cgi
    </Directory>
</VirtualHost>
```

Sekali lagi, *GitWeb* dapat disajikan dengan *CGI* setiap *web server* yang mampu, jika anda memilih untuk menggunakan sesuatu yang lain, seharusnya tidak sulit untuk mengatur. Pada bagian ini. Anda harus dapat mengunjungi `http://gitserver/` untuk melihat repositori *online*, dan anda dapat menggunakan `http://git.gitserver/` untuk meng-*cloning* dan *fetching* repositori anda melalui *HTTP*.

g. *Gitosis*

Menjaga semua pengguna *public key* di dalam *file* `authorized_key` agar akses bekerja dengan baik untuk sementara waktu. Bila anda memiliki ratusan pengguna, itu jauh lebih rumit untuk mengelola proses tersebut anda harus *shell* ke server setiap kali, dan tidak ada kontrol akses, semua orang di berkas ini telah mengakses *read* and *write* ke setiap proyek.

Pada bagian ini, anda mungkin ingin beralih ke proyek *software* yang digunakan secara luas yang disebut *gitosis*. *Gitosis* pada dasarnya adalah satu *set* skrip yang membantu anda mengelola *file* `authorized_keys` serta mengimplementasikan beberapa kontrol akses sederhana. Bagian yang paling menarik adalah bahwa *UI* alat ini untuk menambahkan orang dan menentukan akses yang bukan merupakan antarmuka *web* tetapi khusus repositori *Git*.

Instalasi *gitosis* bukanlah tugas sederhana, tetapi juga tidak terlalu sulit. Ini paling mudah untuk menggunakan *linux server* – contoh berikut menggunkan *ubuntu 8.10 server*.

Gitosis membutuhkan beberapa alat *python*, jadi pertama anda harus menginstal paket *setuptools python*, yang mana ubuntu menyediakan sebagai *python-setuptool.*

```
$ apt-get install python-setuptools
```

Selanjutnya, anda meng-*cloning* dan menginstal *gitosis* dari situs utama proyek:

```
$ git clone git://eagain.net/gitosis.git
$ cd gitosis
$ sudo python setup.py install
```

Itu menginstall beberapa *executable* yang *gitosis* akan menggunakannya. *Gitosis* ingin menempatkan repositori di `/home/git/`, yang benar saja. Tapi anda telah menyiapkan repostitori anda di `/opt/git/`, jadi tidak perlu mngkonfigurasi ulang semuanya, anda cukup membuat *symlinks*:

```
$ ln –s /opt/git /hom/git/repositories
```

Gitosis akan mengelola *key* anda, sehingga anda perlu menghapus *file* saat ini. Kembali menambahkan *key* kemudian, dan biarkan *gitosis* mengontrol *file* `authorized_keys` secara otomatis. Untuk saat ini, memindahkan *file* `authorized_keys` keluar alur :

```
$ mv /home/git/.ssh/authorized_keys \
  /home/git/.ssh/ak.bak
```

Selanjutnya anda harus mengubah *shell* anda kembali untuk user 'git', jika anda mengubah ke perintah `git-shell`. Orang – orang

tetap tidak dapat *log in*, karena *gitosis* yang mengontrol untuk anda. jadi. Mari kita ubah baris ini di file `/etc/passwd` anda:

```
git:x:1000:1000::/home/git:/usr/bin/git-shell
```

kembalikan ke ini:

```
git:x:1000:1000::/home/git:/bin/sh
```

Sekarang saatnya menginisialisasi *gitosis*. Caranya dengan menjalankan perintah `gitosis-init` dengan kunci publik pribadi anda. jika kunci anda tidak diserver anda harus menyalinnya :

```
$ sudo -H -u git gitosis-init < /tmp/id_dsa.pub
Initialized empty Git repository in
/opt/git/gitosis-admin.git/
Reinitialized existing Git repository in
/opt/git/gitosis-admin.git/
```

Hal ini memungkinkan pengguna dengan *key* yang memodifikasi repositori *Git* utama yang mengontrol *setup gitosis*. Selanjutnya, anda harus secara manual mengatur *bit* eksekusi pada skrip `post-update` untuk repositori control baru anda:

```
$ sudo chmod 755 /opt/git/gitosis- \
admin.git/hooks/post-update
```

Jika anda mngatur dengan benar, anda bisa mencoba untuk *SSH* ke server sebagai pengguna yang menambahkan kunci publik untuk menginisialisasi *gitosis*. Anda akan melihat sesuatu seperti ini:

```
$ ssh git@gitserver
PTY allocation request failed on channel 0
fatal: unrecognized command 'gitosis-serve
schacon@quaternion'
Connection to gitserver closed.
```

Itu berarti *gitosis* mengakui anda, tetapi anda menutup karena anda tidak mencoba untuk melakukan perintah *Git*, jadi, mari kita

melakukan perintah *Git* yang sebenarnya – anda akan meng-*cloning* repositori kontrol *gitosis*:

```
# on your local computer
$ git clone git@gitserver:gitosis-admin.git
```

Sekarang anda memiliki sebuah direktori benama `gitosis-admin`, yang memiliki dua bagian utama :

```
$ cd gitosis-admin
$ find .
./gitosis.conf
./keydir
./keydir/scott.pub
```

File `gitosis.conf` adalah *file* kontrol yang anda gunakan untuk menentukan pengguna, repositori, dan perizinan. Direktori `keydir` dimana anda menyimpan kunci publik dari semua pengguna yang memiliki segala macam akses ke repositori anda – satu *file* per-pengguna.

Nama *file* dalam `keydir` (dalam contoh sebelumnya, `scott.pub`) akan berbeda untuk anda – *gitosis* mengamabil nama dari deskripsi pada akhir kunci publik yang diimpor dengan skrip

```
gitosis-init.
$ cat gitosis.conf
[gitosis]
[group gitosis-admin]
writable = gitosis-admin
members = scott
```

Ini menunjukkan anda bahwa user 'scott' – user dengan kunci publik yang diinisialisasi gitosis adalah satu-satunya yang memiliki akses ke proyek `gitosis-admin`.

h. *Git Daemon*

Untuk publik, otentikasi aksesnya *read* ke proyek anda, anda ingin menjalankan lewat protokol *HTTP* dan mulai menggunakan protokol *Git*. Alasan utamanya adalah kecepatan. Protokol git jauh lebih efisisen dan dengan demikian lebih cepat dari protokol *HTTP*. Sehingga dengan itu akan menghemat waktu pengguna.

Sekali lagi, ini adalah untuk yang tidak berkepentingan mengakses *read-only*. Jika anda menjalankan ini pada *server* diluar *firewall* anda, itu hanya digunakan untuk proyek – proyek yang umum didunia. Jika server anda menjalankannya di dalam *firewall* anda, anda mungkin menggunakannya untuk proyek – proyek yang sejumlah besar orang atau komputer (integrasi berkesinambungan atau membangun *server*) akses telah *read-only*, bila anda tidak ingin memiliki untuk menambahkan kunci *SSH* untuk masing-masing.

Dalam kasus apapun protokol *Git* relatif mudah untuk mengatur. Pada dasarnya, anda perlu menjalankan perintah ini dengan cara *daemonized*:

```
git daemon --reuseaddr --base-path=/opt/git/ \
/opt/git/ --reuseraddr
```

Memungkinkan *server* untuk me-*restart* tanpa menunggu waktu koneksi lama keluar, opsi `--base-path` memungkinkan orang untuk meng-*cloning* proyek tanpa menentukan seluruh aturan, dan jalan diakhir memberitahu *Git Daemon* mana untuk mencari repositori yang diekspor. Jika anda menjalankan *firewall*, anda juga

harus membuat lubang didalamnya di port 9418 pada kotak pengaturan ini.

Anda dapat me-*daemonize* proses ini dengan beberapa cara, tergantung pada sistem operasi yang anda jalankan, pada mesin ubuntu, anda menggunkan *script upstart*. Jadi dalam *file* berikut :

```
/etc/event.d/local-git-daemon
```

Anda letakkan skrip ini :

```
start on startup
stop on shutdown
exec /usr/bin/git daemon \
        --user=git --group=git \
        --reuseaddr \
    --base-path=/opt/git/ \
    /opt/git/
respawn
```

Untuk alasan keamanan, sangat dianjurkan untuk memiliki daemon ini dijalankan sebagai pengguna dengan hak akses *read-only* ke repositori anda dapat dengan mudah melakukan ini dengan menciptakan *user* 'git-ro' baru dan menjalankan *daemon* seperti mereka. untuk kesederhanaan kita akan jalankan sebagai pengguna yang sama 'git' yang *gitosis* berjalan diatasnya.

Ketika anda me-*restart* komputer, *git daemon* anda akan berjalan secara otomatis dan *respawn* jika *down*. Untuk menjalankan tanpa harus *reboot*, anda dapat menjalankan ini:

```
initctl start local-git-daemon
```

i. *Hosted Git*

Jika anda tidak ingin melakukan semua pekerjaan yang terlibat dalam menyiapkan *server Git* anda sendiri, anda memiliki beberapa

pilihan untuk *hosting* proyek *Git* anda di situs *hosting dedicated* eksternal. situs *hosting* umumnya cepat dalam mengatur dan mudah untuk memulai proyek – proyek, dan tidak ada pemeliharaan *server* atau *monitoring* yang terlibat. Bahkan jika anda membuat dan menjalankan *server* anda sendiri secara internal, anda mungkin masing ingin menggunakan *situs hosting* publik untuk kode sumber terbuka – itu umumnya lebih mudah untuk komunitas *open source* untuk menemukan dan membantu anda denganya.

Anda memiliki sejumlah besar pilihan untuk memilih dari masing – masing *hosting* yang memiliki kelebihan dan kekurangan yang berbeda. Untuk melihat daftar *up-to-date*, periksa halaman *GitHosting* pada *Git Wiki* utama:

http://git.or.cz/gitwiki/GitHosting

C. Peningkatan Pembelajaran Mahasiswa

1. Pengertian

   a. Peningkatan Pembelajaran

      Strategi untuk meningkatkan pembelajaran mahasiswa, sebagaimana Linda Suskie (2002:1) merumuskannya, yaitu bukti bahwa mahasiswa belajar paling efektif ketika:

      1) Mereka memahami program dan sasaran program dan karakteristik pekerjaan dengan baik.

2) Mereka secara akademi ditantang dan didorong untuk fokus pada pengembangan ketrampilan berfikir tingkat tinggi, seperti berpikir kritis dan pemacahan masalah, serta disiplin pengetahuan lebih khusus.
3) Mereka menghabiskan lebih banyak waktu aktif terlibat dalam pembelajaran dan sedikit waktu untuk mendengarkan kuliah.
4) Mereka terlibat dalam multidimensi tugas "dunia nyata".
5) Mereka memiliki interaksi positif dengan fakultas dan bekerja secara kolaboratif dengan sesama mahasiswa; semua peserta didik – mahasiswa dan dosen – menghormati dan menghargai orang lain sebagai peserta didik.
6) Mereka berpartisipasi dalam kegiatan luar kelas, seperti kegiatan ko-kurikuler dan layanan kesempatan belajar, yang membangun apa yang mereka pelajari di kelas.
7) Tugas dan penelitian yang terkait dengan kegiatan belajar dan fokus pada program dan sasaran program yang paling penting.
8) Mereka memiliki kesempatan untuk merevisi karya mereka.
9) Mereka merefleksikan apa dan bagaimana mereka telah belajar.
10) Mereka memiliki pengalaman yang mumpuni, seperti seminar, magang, studi independen, proyek penelitian, atau tesis, yang memungkinkan mereka mensintesis apa yang telah mereka pelajari selama pengalaman kuliah mereka.

Jadi, pembelajaran mahasiswa dikatakan meningkat apabila mahasiswa, dosen dan perguruan tinggi dapat menerapkan beberapa strategi diatas.

b. Mahasiswa

Mahasiswa adalah peserta didik pada jenjang pendidikan tinggi. Mahasiswa sebagai anggota sivitas akademika diposisikan sebagai insan dewasa yang memiliki kesadaran sendiri dalam mengembangkan potensi diri di perguruan tinggi untuk menjadi intelektual, ilmuan, praktisi, dan/atau profesional. Mahasiswa mengembangkan bakat, minat, dan kemampuan dirinya melalui kegiatan kokurikuler dan ekstrakulikuler sebagai bagian dari proses pendidikan. (UU No.12,2012: Tentang Pendidikan Tinggi)

2. Peningkatan Pembelajaran Mahasiswa

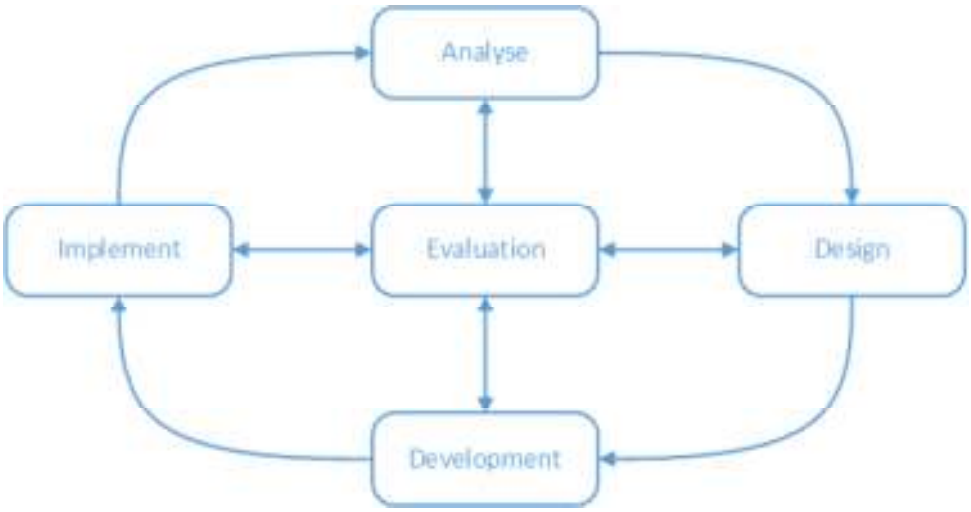
Jadi, mahasiswa di katakan meningkat belajarnya apabila minimalnya mereka berpartisipasi dalam kegiatan luar kelas, seperti kegiatan ko-kurikuler dan layanan kesempatan belajar, yang membangun apa yang mereka pelajari di kelas, mereka menghabiskan lebih banyak waktu aktif terlibat dalam pembelajaran dan sedikit waktu untuk mendengarkan kuliah dan mereka merefleksikan apa dan bagaimana mereka telah belajar. Karena, mahasiswa diposisikan sebagai insan dewasa yang memiliki kesadaran sendiri dalam mengembangkan potensi diri di perguruan tinggi untuk menjadi intelektual, ilmuan, praktisi, dan/atau profesional.

BAB III

METODE PENGEMBANGAN

A. Model Pengembangan

Model pengembangan yang digunakan dalam penelitian ini yaitu menggunakan model *ADDIE,* yang mana model ini digunakan untuk mengembangkan media belajar mata kuliah pemrograman menggunkan *Git Server,* model *ADDIE* digambarkan sebagai berikut :



Gambar 3.1. Model *ADDIE* (Reiser dan Mollenda, 1990)

B. Prosedur Pengembangan

1. Analisis
   a. Analisis Kinerja
      1) Kurangnya partisipasi mahasiswa dalam kegiatan ko-kurikuler memerlukan solusi berupa penyelenggaraan program peningkatan

pembelajaran.

2) Pembelajaran yang masih terpusat pada dosen menyebabkan proses pembelajaran tidak ada peningkatan.
3) Media belajar bersama mahasiswa yang masih individual memerlukan solusi berupa media belajar yang terdistribusi.

b. Analisis Kebutuhan

1) Membuat materi ajar pemrograman sebagai simulasi pembelajaran terdistribusi.
2) Mahasiswa mampu belajar bersama dengan cara terdistribusi menggunakan *Git Server*.
3) Mahasiswa memiliki kesempatan merevisi karya mereka.
4) Rancangan sistem jaringan sebagai media pembelajaran terdistribusi.

c. Analisis Sistem

1) Tinjauan

Tahapan yang diindikasikan akan muncul adalah (1) memanfaatkan *Git Server* sebagai media pembelajaran mahasiswa, (2) Mensosialkan *code* secara terbuka di *intern* mahasiswa, (3) mensosialkan *code* secara terbuka dengan luas dan *global* dengan *GitHub*.

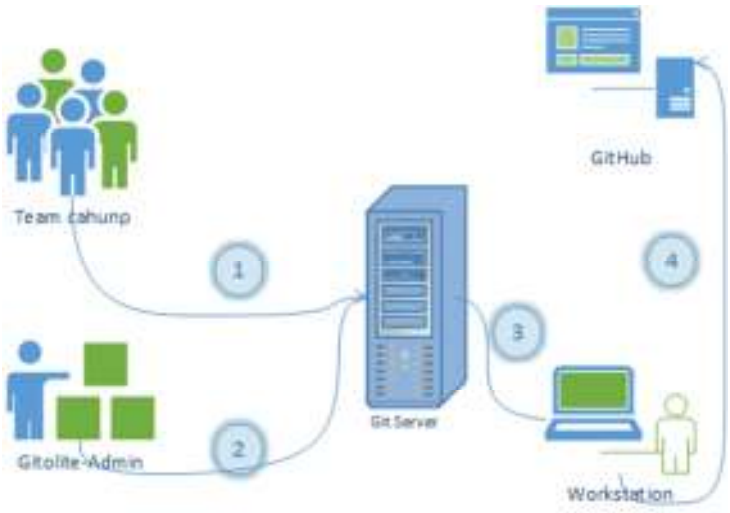
2) Manfaat

Git Server akan mendistribusikan *source code* dari sejumlah mahasiswa yang telah ditentukan dengan metode *social*

coding dengan tahapan yang telah diindikasikan, maka mahasiswa dapat memanfaatkan *Git* Server sebagai (1) pengontrol versi secara terdistribusi pada bahasa pemrograman yang diajarkan di kelas, (2) berkolaborasi kode dengan teman sekelas, (3) bentuk menanamkan sikap berbagi kode secara terbuka dengan mahasiswa lain.

3) Alur Kerja

Alur kerja sistem ini ada empat langkah yaitu (1) Team melakukan *pull* dan *push* ke *Git Server*, (2) *Gitolite-admin* melakukan manajemen *user team*, (3) *Workstation* melakukan sinkronisasi dengan *Git Server* yaitu dengan cara team melakukan *push* dan *pull* ke `git@git.gitserver`, dan (4) *Workstation* bisa melakukan *Pull* dan *Push* ke *GitHub, team* cahunp *forking* ke *GitHub*. Seperti berikut :



Gambar 3.2. *Workflow* Sistem

2. Perancangan

a Perangkat Pembelajaran

1) Tujuan Pembelajaran

(a) Mahasiswa mampu memahami pemrograman berorientasi objek *Inheritance* di *java*.

(b) Mahasiswa mampu memahami fungsi dan kegunaan *Git Server*.

(c) Mahasiswa mampu mengoperasikan *Git Server* sebagai media belajar bersama.

(d) Mahasiswa mampu memahami konsep *social coding*.

(e) Mahasiswa dapat belajar bersama mata kuliah pemrograman dengan terdistribusi.

(f) Mahasiswa mampu mengerjakan tugas mata kuliah pemrograman dengan cara terdistribusi.

(g) Mahasiswa berpartisipasi kegiatan ko-kurikuler.

2) Deskripsi Isi

Karakteristik Pemrograman Berorientasi Objek – *Inheritance*

a) Pendahuluan

Inheritance merupakan bentuk penggunaan kembali perangkat lunak dimana kelas baru dibuat dengan menyerap anggota kelas yang ada dan menghiasi mereka dengan kemampuan baru atau diubah . Dengan *Inheritance*, Anda dapat menghemat waktu selama pengembangan program dengan

mendasarkan kelas baru yang ada. Ini juga meningkatkan kemungkinan bahwa sistem akan diterapkan dan dipelihara secara efektif.

Ketika membuat kelas, sebelum menyatakan anggota benar-benar baru, Anda dapat menetapkan bahwa kelas baru harus mewarisi anggota kelas yang ada. Kelas yang ada disebut *superclass*, dan kelas baru *subclass*. (dalam bahasa pemrograman C++ mengacu pada *superclass* sebagai kelas dasar dan *subclass* sebagai kelas turunan.) Setiap *subclass* bisa menjadi *superclass* untuk *subclass* kedepan.

Sebuah *subclass* dapat menambahkan kolom dan metode sendiri. Oleh karena itu, subclass lebih spesifik daripada *superclass* dan merupakan kelompok obyek yang lebih khusus. *Subclass* menunjukkan perilaku *superclass* dan dapat memodifikasi perilaku mereka sehingga mereka beroperasi tepat untuk *subclass*.

Tabel 3.1. *Superclass* dan *Subclass*

Superclass	Subclasses
Student	GraduateStudent, UndergraduateStudent
Shape	Circle, Triangle, Rectangle, Sphere, Cube
Loan	CarLoan, HomeImprovementLoan, MortgageLoan
Employee	Faculty, Staff
BankAccount	CheckingAccount, SavingsAccount

Setiap objek *subclass* adalah obyek *superclass*-nya, dan satu *superclass* dapat memiliki banyak *subclass*, himpunan benda

diwakili oleh *superclass* sering lebih besar dari set objek yang diwakili oleh salah satu *subclass*. Sebagai contoh, Kendaraan *superclass* mewakili semua kendaraan, termasuk mobil, truk, kapal, sepeda dan sebagainya. Sebaliknya, Mobil *subclass* merupakan kecil, bagian yang lebih spesifik kendaraan.

b) Susunan hirarki anggota komuniats dalam universitas

Hubungan warisan membentuk struktur hirarkis *treelike*. Sebuah *superclass* ada dalam hubungan *hirarki* dengan *subclass*. Mari kita mengembangkan hirarki kelas, juga disebut hierarki *inheritance*. Sebuah komunitas universitas memiliki ribuan anggota, termasuk karyawan, mahasiswa dan alumni. Karyawan baik fakultas ataupun anggota staf fakultas anggota yang baik *administrator* (misalnya, dekan dan ketua departemen) atau guru. Hirarki bisa mengandung banyak kelas-kelas lain. Sebagai contoh, siswa dapat mahasiswa, pascasarjana atau sarjana. Mahasiswa dapat mahasiswa, mahasiswi, junior atau senior.

Gambar 3.3 Hirarki *CommunityMember*

c) *Inheritance*

Suatu *class* dapat mewariskan atribut dan *method* kepada *class* lain (*subclass*), serta membentuk class hierarchy, tantunya penting untuk *Reusability*.

Contoh Program :

```
/*
 * Sepeda.java
 */

public class Sepeda{
  private int gir;
  void setGir(int pertambahanGir) {
        gir= gir+ pertambahanGir;
  }
        int getGir() {
        return gir;
  }
}

/*
 * SepedaGunung.java
 */

public class SepedaGunung extends Sepeda{
private int sadel;
void setSadel (int jumlah) {
  sadel = getGir() - jumlah;
  }

  int getSadel(){
  return sadel;
  }
}

/*
 * SepedaGunungBeraksi.java
 */

public class SepedaGunungBeraksi {
     public static void main(String[] args) {
     SepedaGunung sg=new SepedaGunung();
     sg.setGir(3);
     System.out.println(sg.getGir());
     sg.setSadel(1);
     System.out.println(sg.getSadel());
  }
}
```

3) Pendekatan Pembelajaran

Pendekatan : *Social Coding*

Metode : Kolaborative

Tugas : Membuat Aplikasi sederhana menggunakan *class*, *object*, dan *method*.

Media : Komputer, Internet, *Git Server* (Lokal dan *Hosting*)

4) Evaluasi

Kehadiran, Tugas, UTS, dan UAS

Batas Lulus : 75%, Nilai : 1Kehadiran + 1Tugas + 1UTS + 2 UAS : 5, Kehadiran Minimal : 85% dari perkuliahan.

5) Tabel 3.2 Rincian Materi Perkuliahan Setiap Pertemuan

Pertemuan	**Pokok Pembahasan**
Pertama	Karakteristik PBO Java
Kedua	Instalasi dan Konfigurasi – Git
Ketiga	Latihan Java dengan Git
Keempat	Tugas dan Evaluasi

6) Daftar Buku

Sharon Zakhour et al, *The Java Tutorial Fourth Edition*, http://java.sun.com/docs/books/tutorial

Cay Horstmann, Big Java: *Early Objects 5th Edition*, John Wiley & Sons, 2013

Deitel & Deitel, *Java Howto Program 9th Edition*, Prentice Hall, 2012

b Perangkat Sistem

Pada tahapan ini akan direncanakan rancangan kebutuhan perangkat sistem untuk membangun *Git Server* yaitu kebutuhan *Hardware* dan *Software*.

1) *Software*

Bagian ini akan menjelaskan kebutuhan *software* pada *Server-Side* (*Linux*), *Client-Side* (*Linux*), *Client-Side* (*Windows*), dan *Administrator-Side* (*Windows*) untuk didapatkan rancangan sistem yang sesuai.

a) *Server-Side* (*Linux*)

(1) *OpenSSH-Server*

Penjelasan tentang *OpenSSH* diuraikan Contributor, Team (2012:80-82) seperti berikut:

OpenSSH – versi tersedia secara bebas dari *Secure Shell* (*SSH*) yaitu sebuah alat semisal *protocol* untuk mengendalikan komputer jarak jauh atau mentransfer *file* antar komputer. Sedangkan *OpenSSH-Server* yaitu Aplikasi *protocol Secure Shell* yang melayani akses *SSH* dari klien. Untuk menginstalnya dengan perintah:

```
$ sudo apt-get install openssh-server
```

Direktori *files* konfigurasi *SSH* ada di :

```
$ nano /etc/ssh/sshd_config
```

Aplikasi ini digunakan untuk melayani lalu lintas *git version control* dalam setiap kolaborasi *repository*.

(2) *Git-1.9.2.tar.gz*

Penjelasan mengenai instalasi *Git* dari *source*-nya dijabarkan oleh Schacon (2009:8) sperti berikut:

Menginstal *git* dari *source*-nya, *source code git* dapat di download dari alamat:

https://www.kernel.org/pub/software/scm/git/

Untuk menginstal git dari *source* maka harus memiliki *library* yang dibutuhkan oleh *git* : *curl, zlib, openssl, expat,* dan *libiconv*.

Untuk menginstal *library*-nya dengan perintah :

```
$ apt-get install libcurl4-gnutls-dev
libexpat1-dev gettext libz-dev libssl-
dev build-essential
```

Selanjutnya *compile* dan *install* :

```
$ tar –zxf git-1.9.2.tar.gz
$ cd git-1.9.2
$ make prefix =/usr/local all
$ make prefix =/usr/local install
```

(3) *Gitolite-admin*

Gitolite - lapisan kontrol akses di atas *Git*, berikut adalah fitur yang kebanyakan dapat kita lihat:

(a) Menggunkan satu *user unix* ("*real*" *user*) pada *server*.

(b) Menyediakan akses ke banyak pengguna *gitolite.*

- Mereka bukan pengguna "nyata"
- Jadi pengguna tidak dapat mengakses *shell*.

(c) Mengendalikan akses ke banyak *git repository.*

- akses *read* dikendalikan di tingkat *repo*
- akses *write* dikendalikan ditingkat `branch/tag/file/directory`, termasuk bisa *rewind*, *create*, dan *delete* `branches/tags`.

(d) Dapat diinstal tanpa akses *root*, dengan asumsi *git* dan *perl* sudah diinstal.

(e) Otentikasi ini paling sering dilakukan dengan menggunkan *sshd*.

Untuk menginstalnya *clone* dahulu dari *github*. Dengan asumsi direktori `$HOME/bin` sudah dibuat dan ini adalah jalan agar bisa menginstal gitolite:

```
$ git clone \
git://github.com/sitaramc/gitolite
$ gitolite/install –to /home/git/bin
$ /home/git/bin/gitolite-setup –pk \
  id_rsa_administrator.pub
```

(4) *Gitweb*

Antarmuka *web git* (*web frontend* ke *git repository*), *gitweb* menyediakan antarmuka *web* untuk *Git repository*. Fitur-fiturnya antara lain:

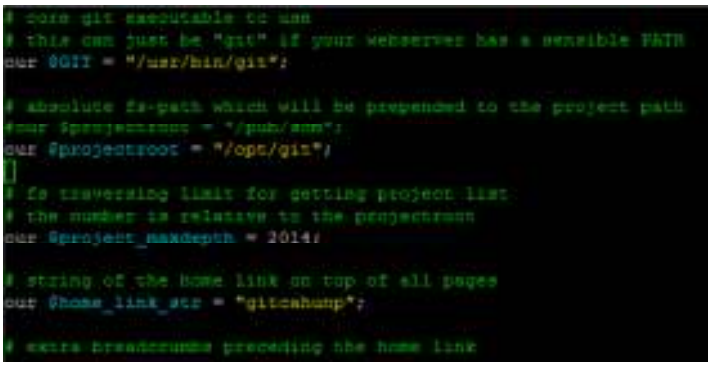
- Melihat beberapa *repositori git* dengan akar umum.
- *Browsing* setiap revisi dari *repository*.
- Melihat isi *file* dalam *repository* pada disetiap revisi.
- Melihat *log* revisi *branch*, sejarah *file* dan *directory*, melihat apa yang ketika diubah, dan oleh siapa.

- Melihat segala sesuatu yang berubah dalam revisi, dan langkah melalui revisi satu per satu, melihat sejarah *repsitory*.

Untuk menginstal dari *source*-nya akan dapat versi terbaru :

```
$ cd git-1.9.2
$ cp –Rf gitweb/ /var/www/
$ nano /var/www/gitweb/gitweb.cgi
```

Selanjutnya rubah `$projectroot` *repository* di tempatkan kemudian simpan, sedikit konfigurasi *DNS*, agar dapat diakses dengan alamat *domain* (misalnya : `http://www.gitserver`) dan tentu *web server* sudah terinstal pada komputer *server*.



Gambar 3.4 Komfigurasi *gitweb.cgi*

(5) *Git-daemon*

Sebuah *server* benar – benar sederhana untuk git *repository*, benar – benar sederhana *TCP git daemon* yang bisaanya *listen* pada port “DEFAULT_GIT_PORT” *alias*

9418. Ini menunggu untuk koneksi meminta layanan, dan akan melayani *rute* yang jika diaktifkan.

Ini dibuktikan pada direktori yang memiliki *file* "*git-daemon-eksport-ok*", dan akan menolak untuk mengekspor setiap *direktory Git* yang belum secara explisit telah ditandai untuk di ekspor dengan cara ini (yaitu kecuali parameter *–export-all* di tentukan), jika anda melewati beberapa jalur *direktory* menggunakan *git daemon*, anda dapat lebih membatasi daftar pengguna.

Mengaktifkan *git-daemon* :

```
$ git daemon --base-path=/opt/git --detach-
syslog --export-all
```

(6) *Bind9*

Bind9 - aplikasi yang digunakan *server* untuk membangun *DNS server*, menempatkan *server DNS* pada jaringan memungkinkan untuk penggantian alamat *ip address* mesin *server* dengan nama yang mudah diingat, *ip address Git Server* 192.168.92.1 dan 192.168.91.1 menjadi alamat *domain* `http://www.gitserver` dan `http://www.git.gitserver`.

Sebeneranya peratama kali menginstal *ubuntu server 12.04.1 LTS bind9* (*DNS server*) dan *apache2* (*web server*) dapat diinstal bersamaan ketika menginstal sistem operasinya.

b) *Client-Side* (*Linux*)

(1) *OpenSSH-client*

OpenSSH adalah versi tersedia secara bebas dari *Secure Shell* (*SSH*) yaitu sebuah alat semisal *protocol* untuk mengendalikan komputer jarak jauh atau mentransfer *file* antar komputer. Sedangkan *OpenSSH-client* merupakan aplikasi *protocol Secure Shell* yang digunakan untuk melakukan *generating SSH-Keygen*.

Untuk menginstalnya dengan perintah, bisaanya ketika menginstall *linux openSSH-client* sudah terinstal :

```
$ sudo apt-get install openssh-client
```

Direktori *file* konfigurasi *SSH* ada di :

```
$ nano /etc/ssh/sshd_config
```

(2) *Git-core*

Git-core – menginstal *git* dari *binary installer*-nya yaitu nama *package* aplikasi *git* yang diinstal pada sistem operasi *linux*, kalau di *windows* menggunakan *Msysgit* keduanya sama-sama aplikasi *git*, untuk melakukan perintah instalasi pada *linux* keluarga berbasis *debian* menggunkan perintah :

```
$ apt-get install git-core
```

Untuk linux berbasis *fedora* :

```
$ yum install git-core
```

c) *Client-Side* (*Windows*)

(1) *Msysgit*

Msysgit – lingkungan pengembangan untuk mengkompilasi *git* untuk *windows*, merupakan *installer* yang menginstal *Git* dan hanya *git*. Versi *installer git* dapat di download di alamat http://msysgit.github.io, kemudian jalankan untuk proses instalasi.

d) *Administrator-Side* (*Windows*)

(1) *Msysgit*

Pada sisi *administrator git* digunakan untuk mengkonfigurasi *gitolite-admin*.

(2) *Putty*

Putty – sebuah antar muka grafis yang populer untuk *SSH client* pada *windows*, bisa *download* pad alamat:

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html.

2) *Hardware*

Pada bagian ini menjelaskan spesifikasi *hardware server* yang digunakan dan spesifikasi minimum untuk *client* dapat menginstal *git*.

a) Spesifikasi *Server-Side* (*Linux*)

Spesifikasi *server* yang digunakan dapat di tampilkan

menggunakan perintah `$ dmidecode` :

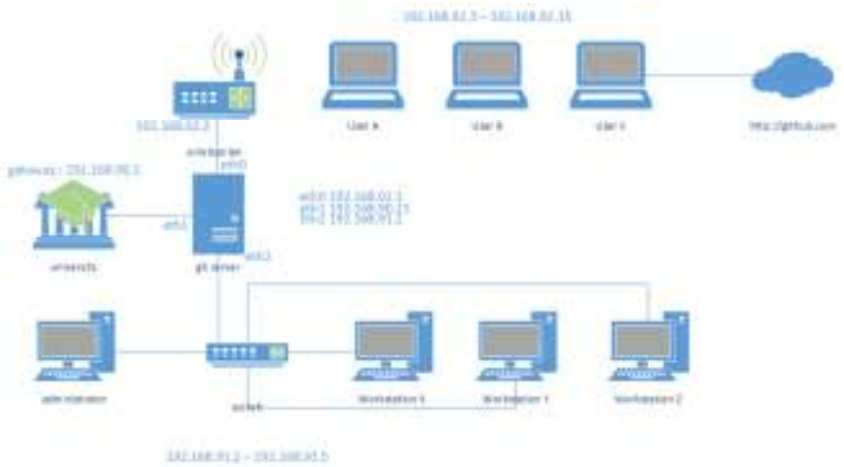
```
#dmidecode 2.11
SMBIOS 2.4 present.
49 structures occupying 1953 bytes.
Table at 0x000E84B0.

BIOS Information
        Vendor: Intel Corp.
        Version: RQG4110H.86A.0009.2009.0108.1005
        Release Date: 01/08/2009
Base Board Information
        Manufacturer: Intel Corporation
        Product Name: DG41RQ
        Version: AAE54511-203
        Serial Number: AZRQ928004DU
Processor Information
        Socket Designation: LGA775
        Type: Central Processor
        Family: Quad-Core Xeon 5300
        Manufacturer: Intel(R) Corp.
        ID: 7A 06 01 00 FF FB EB BF
        Signature: Type 0, Family 6, Model 23,
Stepping 10
    Version: Pentium(R) Dual-Core CPU      E5300
@ 2.60GHz
Memory Array Mapped Address
        Starting Address: 0x00000000000
        Ending Address: 0x0007FFFFFFF
        Range Size: 2 GB
        Physical Array Handle: 0x002A
        Partition Width: 1
```

b) Spesifikasi *client-side*

Spesifikasi minimum untuk *client* agar dapat menjalankan *git* yaitu mengikuti sistem operasi yang digunakan, *git* dapat diinstal pada sistem operasi *linux*, *windows* dan *unix*, jadi tergantung dari sistem operasi yang digunakan, akan tetapi kebanyakan *hardware* dan sistem operasi yang digunakan sudah memenuhi spesifikasi minimum sehingga bisa melakukan instalasi *git* tanpa ada permasalahan.

3) Jaringan

Jaringan yang dibangun untuk implementasi *Git Server* menggunakan konsep menggabung jaringan lokal dan *Internet* menggunkan topologi star dengan desain rancangan jaringan seperti berikut :



Gambar 3.5 Pengalamatan *Ip Address* Jaringan

Pengalamatan *ip address* setiap platform:

a) *Server-side*

Menambahkan *nameserver* pada `/etc/resolv.conf`

```
nameserver 192.168.91.1
nameserver 192.168.92.1
nameserver 8.8.8.8
nameserver 8.8.4.4
```

Menambahkan *ip address* pada *Ethernet* `eth0` yang terhubung ke *wirelessLAN* :

```
$ nano /etc/network/interfaces
auto eth0
iface eth0 inet static
        address     192.168.92.1
        netmask     255.255.255.240
```

Ethernet `eth1` yang terhubung *gateway* `192.168.90.1` :

```
auto eth1
iface eth1 inet static
        address 192.168.90.21
        netmask 255.255.255.0
        gateway 192.168.90.1
```

Ethernet `eth2` yang terhubung *switch* :

```
auto eth2
Iface eth2 inet static
        ip address 192.168.91.1
        netmask 255.255.255.248
```

b) *Client-Side*

Ip Address pada komputer *administrator* dan *workstation* X, Y, dan Z dapat menggunakan rentang *ip address* 192.168.91.2 – 192.168.91.5, *netmask* 255.255.255.248, *gateway* 192.168.91.1, *DNS* 192.168.91.1 dan 192.168.92.1.

Ip address pada *User* A , B, dan C mendapatkan *ip address* secara *otomatis* dari *wirelessLAN* dengan rentang *ip address* 192.168.92.3 – 192.168.92.15, *netmask* 255.255.255.240, *gateway* 192.168.92.2, hanya sedikit melakukan perubahan manual pada *DNS* yaitu 192.168.91.1 dan 192.168.92.1 pemberian *DNS* tersebut agar dapat mengakses alamat `http://www.gitserver` dan `git@git.gitserver`. *Git Server* dapat akses *internet* dari *gateway* universitas yaitu 192.168.90.1.

Pada gambar di atas bahwa *user* A , B, dan C dapat melakukan koneksi ke http://github.com ketika terhubung internet sehingga *user* dapat *pull* dan *push* ke *repository upstream* yang telah disiapkan.

3. Pengembangan

a. Materi

1) Pemrograman Java *OOP*

Pemrograman yang diajarkan dikelas yaitu java *OOP* yang meliputi *Abstraction*, *Encapsulation*, *Inheritance*, dan *Polymorphism*.

2) Kolaboratif

Materi yang di ajarkan dikombinasikan dengan media *source code* management *system* yang dikenal dengan *system* terdistribusi berbasis *Git Server*, materi berupa *power point* kemudian latihan dan tugas dikerjakan dengan cara kolaborasi dengan teman – teman sekelompok.

Untuk mengembangkan dan mengimplementasikan materi yang diajarkan kelas bisa bergabung dan berkolaborasi dengan mahasiswa lain yang sama–sama menggunakan *Git Server*, sehingga mahasiswa memiliki langkah nyata untuk mengembangkan bakat dan kemampuannya.

b. Media

1) *Software*

Software yang dibutuhkan untuk *local repository* menggunkan *msysgit* untuk *windows, git-core* untuk *linux*, sedangkan untuk system terpusat yang biasa juga dikatakan *bare repository* sebagai kolaborasi menggunakan *Git Server* yang dibangun menggunakan *server linux* dan kedua menggunkan github.com sebagai *hosting repository online* berbasis *Git*.

2) *Hardware*

Hardware yang dapat digunakan adalah sebuah laptop dan komputer *server* sebagai *bare repository* yang digunakan untuk *system* terpusat ketika mahasiswa mengerjakan latihan dan tugas mata kuliah pemrograman.

4. Implementasi

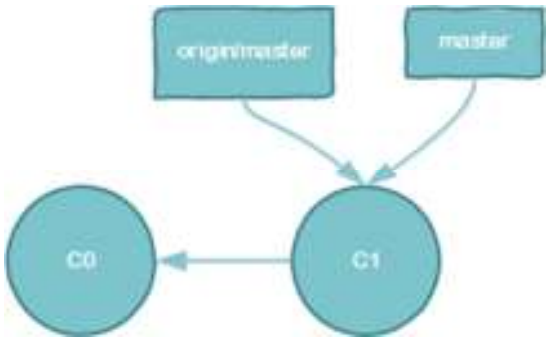
Pada bagian ini menguraikan desain *input*, *output*, dan *process* untuk melakukan penjabaran sistem *git*, yaitu menjelaskan substansi sistem yang paling mendasar hingga kolaborasi proyek dengan tim, yaitu menampilkan proses kolaborasi proyek *trafficLight,* yaitu sebagai contoh *author* dengan nama admincahunp, John, dan Aliece.

a. Desain *Input*

Pada subbab ini akan menguraikan desain *input* dari setiap perintah yang paling mendasar dan kolaborasi pada *Git*.

1) Perintah `$ git init`

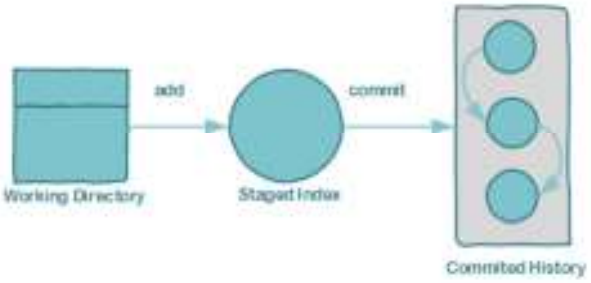
Menjalankan `$ git init` membuat sebuat subdirektori `.git`, pada *root* proyek (direktori proyek), yang berisi semua metadata yang diperlukan untuk *repo*.



Gambar 3.6 Diagram *Git Init*

2) Perintah `$ git add` dan `$ git commit`

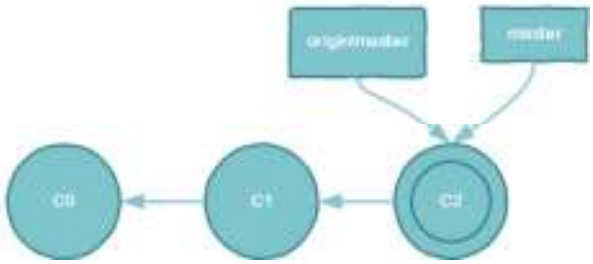
Menjalakan `$ git add` merupakan perintah menambahkan perubahan dalam *working direktory* ke *staging index*. Memberitahu *git* bahwa ini akan menyertakan *update* ke *file* tertentu yang selanjutnya di *commit*. `$ git add` tidak benar-benar merubah *repository* dan perubahan benar-benar tercatat sampai menjalankan `$ git commit`.



Gambar 3.7. Diagram *Git Add* dan *Git Commit*

3) Perintah `$ git commit`.

Sedangakan `$ git commit` perintah untuk melakukan *snapshot staged* untuk melakukan *history project* (*update repository*).



Gambar 3.8. Diagram *History Git Commit*

4) Perintah `$ git status` dan `$ git log`.

`$ git status` menampilkan keadaan *working direktory* dan *staged area*, ini memungkinkan dapat melihat perubahan yang dipentaskan, keluaran status tidak menunjukkan informasi mengenai *history project* yang dilakukan. Untuk menampilkan *history project* yaitu menggunkan perintah `$ git log`.

Gambar 3.9. Diagram *Git Log*

5) Perintah `$ git clone.`

Perintah `$ git clone` merupakan salinan repositori *git* yang sudah ada, kecuali *working copy* merupakan *git* penuh repositori, memiliki sejarah sendiri, mengelola *file* sendiri dan lingkungan benar – benar terisolasi dari repositori aslinya.

Gambar 3.10. Diagram *Git Clone*

6) Perintah `$ git config.`

Perintah `$ git config` melakukan konfigurasi instalasi *git* (repositori individu) dari baris perintah. Perintah ini dapat menentukan segala sesuatu dari info pengguna yang menampilkan setiap perilaku repositori.

a) Konfigurasi *Global* untuk pengaturan spesifik pengguna `~/.gitconfig`

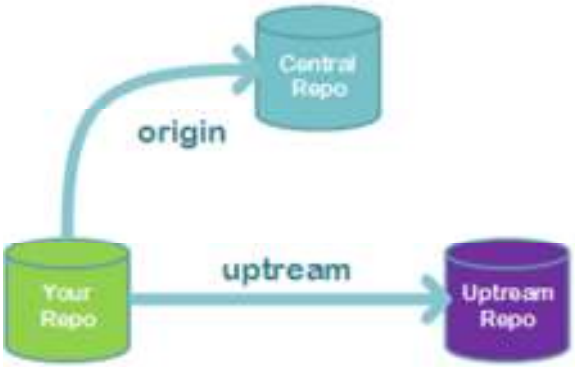
```
$ git config –global [key] [value]
```

b) Konfigurasi *Local* untuk pengaturan spesifik repositori `<repo>/.git/config`

```
$ git config [key] [value]
```

7) Perintah `$ git remote`

Perintah `$ git remote` memungkinkan membuat, melihat, dan menghapus koneksi ke-repositori lain. Koneksi *remote* seperti *bookmark link* langsung ke repositoi lain. Menyediakan kases *real-time* ke repositori lain.



Gambar 3.11 Diagram *Git Remote*

Contoh :

```
$ git remote –v
$ git remote add <alias> <url>
$ git remote rm <alias>
$ git remote rename <alias-lama>  <alias-baru>
```

Ketika melakukan `$ git clone` maka *bookmark url*-nya *central repo* sedangkan `<name-alias>` yang terdaftar adalah *origin*, apabila akan menambahkan *remote* ke *upstream* dengan perintah :

```
$ git remote add upstream \
http://git.gitserver/upstream.git
```

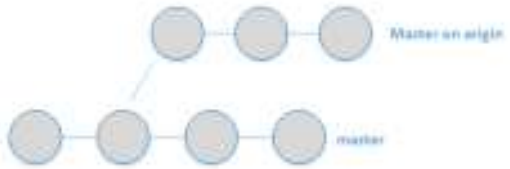
8) Perintah `$ git pull` dan `$ git push`

`$ git pull` proses mengambil dari dan menggabung

dengan repositori lain atau cabang lokal.

a) Sebelum *pulling*.

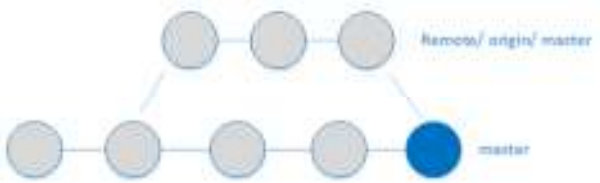
Sebelum proses *pull* dilakukan maka kondisi *master repository* yang ada pada *origin* berbeda dengan *local master,* perbedaannya adalah yang terakhir kali di *pull* sampai perubahan terbaru.



Gambar 3.12. Sebelum *Pulling*

b) Setelah *pulling*.

Kondisi setelah *pull,* maka *repository origin master* menjadi sama dengan *local master*.



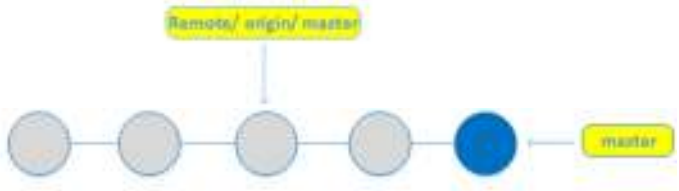
Gambar 3.13. Setelah *Pulling*

`$ git push` proses *update remote refs* dengan objek – objek terkait.

a) Sebelum *Pushing*.

Push berlawanan dengan proses *pull*, kondisi sebelum

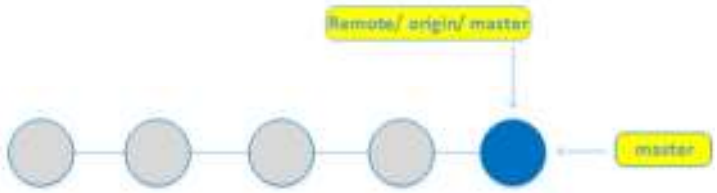
push pada gambar 3.14 menunjukkan *local master* lebih *update* dari pada *origin master*, yang selanjutnya dilakukan *push.*



Gambar 3.14 Sebelum *Pushing*

b) Setelah *pushing*.

Setelah *push* dilakukan maka kondisi menjadi sama antara *origin master* dengan *local master*, yaitu *origin master* menjadi *update* menyesuaikan *local master*.



Gambar 3.15 Setelah *Pushing*

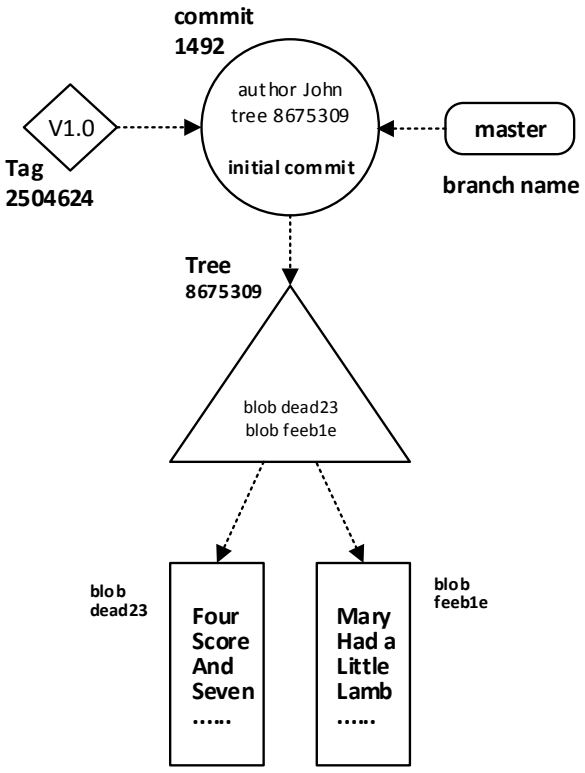
Intinya peruahan *repository* yang ada pada *local master* dengan *origin master* disamakan dengan perintah `$ git pull` dan `$ git push`.

b. Desain *Output*

Pada bagian ini menjabarkan desain *output* sistem *git* ketika

melakukan perintah dasar, saya rasa cukup mengetahui desain *output* sistem *Git* dalam perintah dasar saja tidak perlu kolaborasi, untuk kolaborasi nanti akan di bahas pada desain proses, tidak jauh beda dengan desain *input* hanya saja desain *output* lebih sulit untuk dijabarkan.

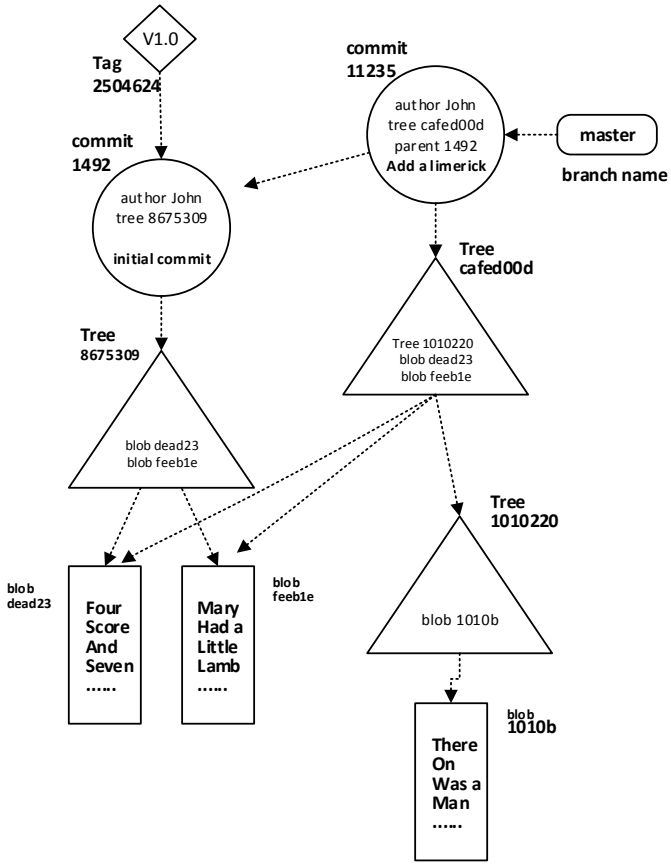
Desain *output Git objects* berikut menguraikan bagaimana sistem *Git* mengolah *files project*, desain ini menunjukkan keadaan *repository* setelah sekali mengawali *commit* dengan menambahkan dua *files*, kedua *file* tersebut di direktori tingkat atas, kedua *file* tersebut berada pada cabang *master*, dan nama *tag* V1.0, kemudian menunjuk ke *commit* dengan *ID* 8675309.



Gambar 3.16. *Git Objects* (Loeliger, 2012:37)

Sekarang kita tambahkan subdirektori baru dengan satu *file* didalamnya, *object* yang dihasilkan akan terlihat seperti gambar, seperti pada gambar sebelumnya *commit* terbaru telah menambahkan satu hubungan *object tree* untuk mewakili kondisi seluruh struktur direktori dan *file*.

Pada gambar ini, *object tree* dengan ID cafed00d.



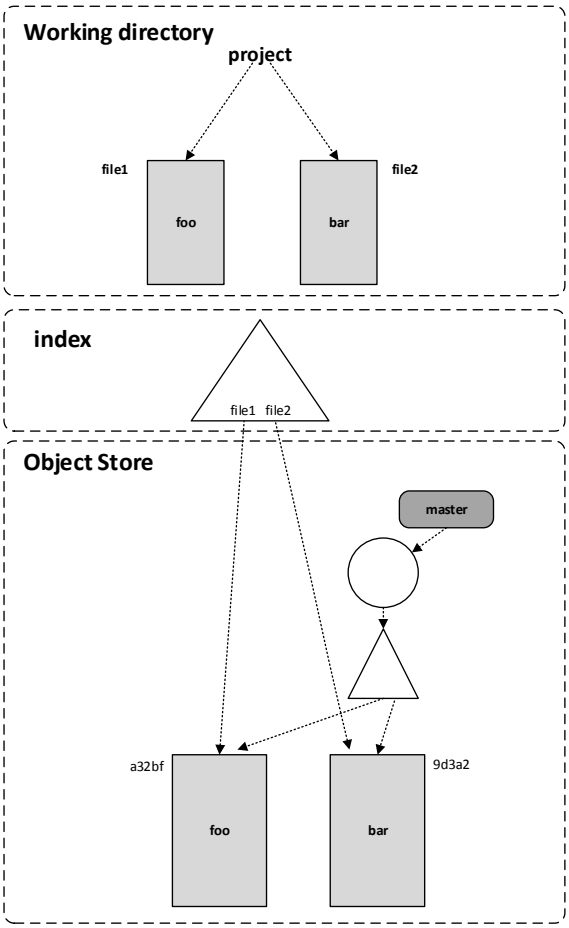
Gambar 3.17. *Git Objects after Second Commit*. (Loeliger, 2012:38)

Karena direktori tingkat atas mengalami perubahan dengan penambahan subdirektori, isi *object tree* tingkat atas berubah juga,

sehingga git memperkenalkan *tree* baru cafed00d.

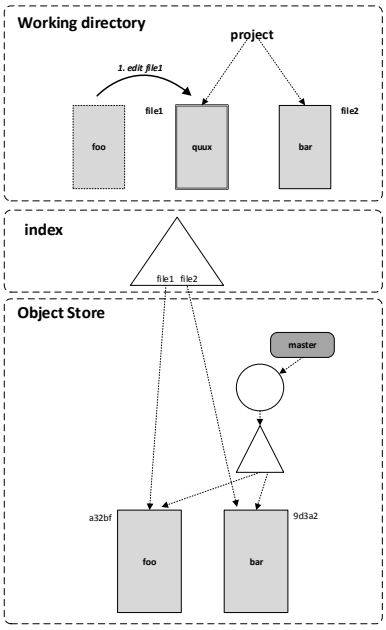
Namun, *blob* dead23 dan feeb1e tidak berubah dari *commit* pertama dan kedua, git menyadari bahwa ID tidak berubah dan dengan demikian dapat langsung dirujuk dan dimiliki oleh *tree* cafed00d.

Untuk mendalami gambar *git object* sebelumnya, kita ulas secara detail tiga kondisi yaitu *working directory*, *index*, dan *object store* (yang ditunjukkan oleh *HEAD* dari *master HEAD*). Gambar–gambar berikut disinkronkan secara berurutan.



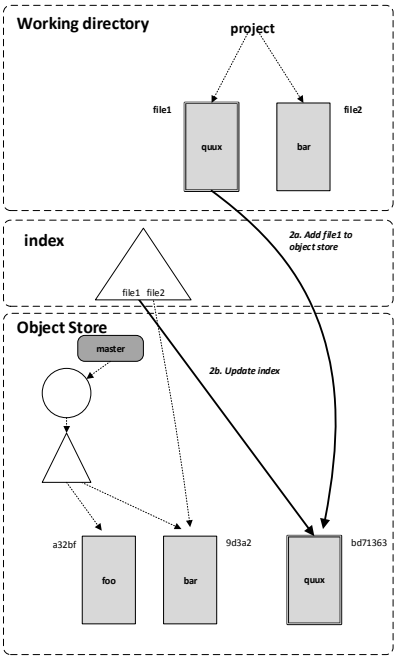
Gambar 3.18 *Inital Files* dan *Objects* (Loeliger, 2012:61)

Selanjutnya gambar setelah *editing file1*.



Gambar 3.19. Setelah *Editing File1* (Loeliger, 2012:61)

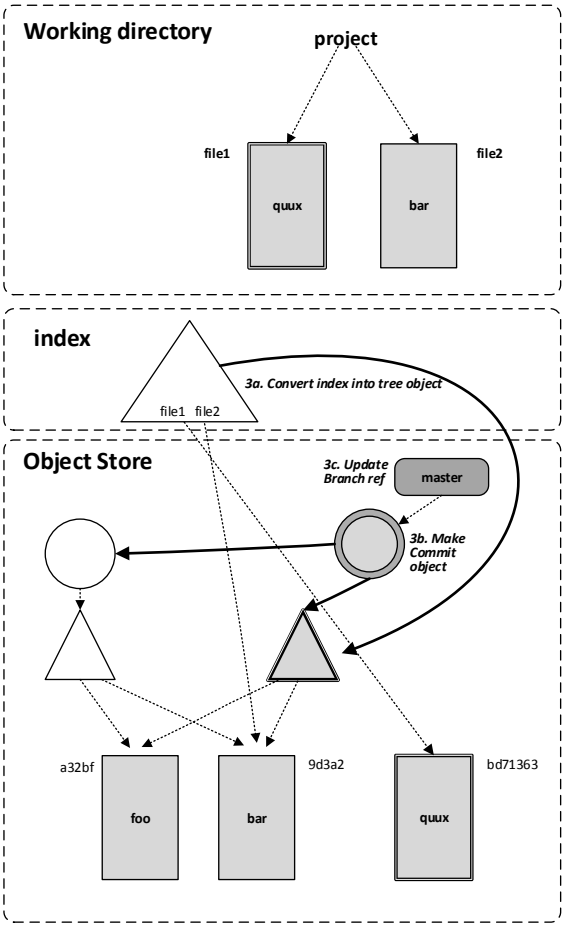
Setelah *git add*.



Gambar 3.20 Setelah *Git Add* (Loeliger, 2012:63)

Selanjutnya setelah *git commit*.

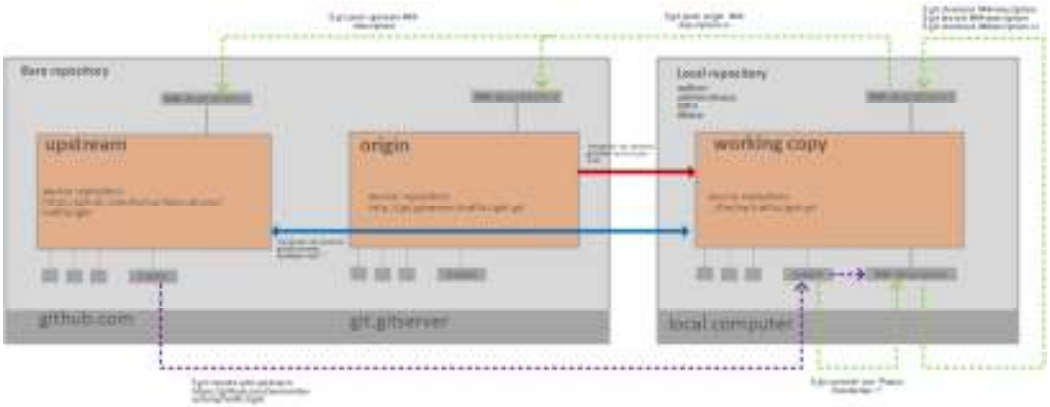
Gambar berikut menujukkan, mengawali *commit* dengan tiga langkah. Pertama, virtual *object tree* yang mana *index* akan diconversi menjadi *object tree* nyata dan ditempatkan kedalam *object store* dibawah nama *SHA*-nya. Kedua, sebuah *object commit* baru dibuat dengan pesan *log* anda. *Commit* baru menunjuk ke *object tree* terbaru dibuat dan juga dengan sebelumnya atau *parent commit*. ketiga, *ref* cabang *master* dipindahkan dari *commit* terbaru ke *commit object* yang baru dibuat, menjadi *master HEAD*.



Gambar 3.21. Setelah *Git Commit* (Loeliger, 2012:64)

c. Desain Proses

Langkah ini mensimulasikan *project* bernama *trafficLight*, dengan tiga *author* yaitu admincahunp, John, dan Aliece sebagai *local computer*, *git server* memanfaatkan dua *remote alias* yaitu git.gitserver sebagai *remote origin* dan github.com sebagai *remote upstream*.



Gambar 3.22 Desain Model Simulasi

1) Langkah pertama,

Author admincahunp membuat *project trafficLight* kemudian melakukan `$ git push origin master` akan tetapi sebelumnya, menambahkan *remote alias* dengan perintah `$ git remote add origin git@git.gitserver:trafficLight.git`

2) Langkah kedua,

Author John melakukan *clone* dari *remote origin master* dengan perintah :

```
$ git clone git@git.gitserver:trafficLight.git
```

setelah melakukan *clone* author John melakukan sedikit modifikasi pada *project trafficLight.*

3) Langkah ketiga,

Author john melakukan *remote alias upstream* dengan perintah `$ git remote add upstream git@github.com:komunitas-cahunp/trafficLight.git` kemudian melakukan *pushing* ke *upstream* dengan perintah `$ git push upstream master`.

4) Langkah keempat,

Author admincahunp bisa melakukan penambahan *remote alias upstream* dengan perintah yang sama seperti yang dilakukan author John dan melakukan *pulling* dan *pushing* kapanpun.

5) Langkah kelima,

Author Aliece bisa melakukan *clone* kedua *git server* yaitu `git.gitserver` maupun github.com dan melakukan *pulling* dan *pushing* kapanpun.

5. Evaluasi

Pada tahapan ini menguraikan tampilan *input*, *output*, dan modul program untuk mengevaluasi sistem, yaitu menjelaskan substansi sistem yang paling mendasar hingga kolaborasi proyek dengan tim, yaitu menampilkan proses kolaborasi proyek *trafficLight,* yaitu sebagai contoh author dengan nama admincahunp, John, dan Aliece.

a. Tampilan *Input*

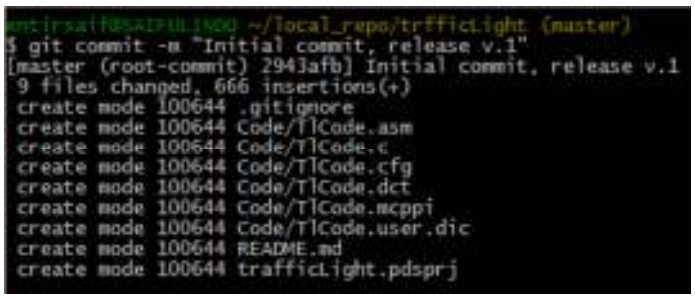
Untuk mengawali pertama kali menciptakan *git repository* menggunakan perintah `$ git init`, sebelumnya membuat repositori proyek dengan nama `<trafficLight>` dengan perintah `$ mkdir trafficLight`.

Gambar 3.23. ***Git Init Shell***

Tampilan input berikutnya yaitu dengan perintah `$ git add .` yaitu untuk menambahkan semua ekstensi *file projects* pada *working tree* ke *staging index*.

Gambar 3.24. ***Git Add Shell***

Setelah tahapan *staging index* selanjutnya perintah `$ git commit –m "Pesan Perubahan"` untuk sistem melakukan rekaman perubahan file proyek yang kemuadian di simpan ke *git repository*.



Gambar 3.25. ***Git Commit Shell***

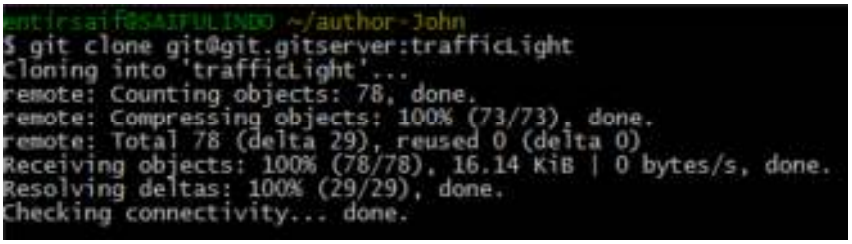
Setelah *files project* sudah selesai dalam pengerjaannya maka siap di *publish* untuk melakukan tahapan kolaborasi sehingga author atau team lain dapat ikut serta mengembangkan *project*, yaitu dengan cara menambahkan *remote alias* kemudian melakukan *push* pada *author* admincahunp, sebelumnya harus mempersiapkan *repository bare* pada `git.gitserver` dengan alamat `git@git.gitserver:trafficLight.git`

Gambar 3.26. ***Git Remote Shell***

Tahapan berikutnya melakukan *push* untuk mengupload *project traficLight* ke `git.gitserver` dengan perintah `$ git push origin master`.

Gambar 3.27. ***Git Push Shell***

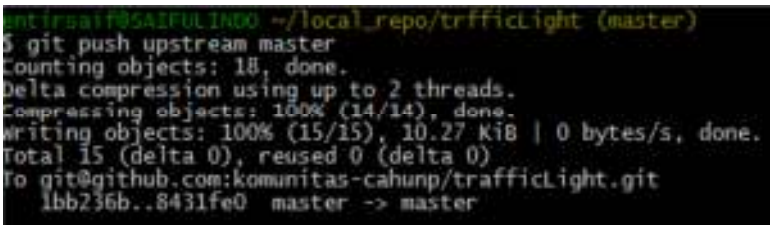
Kemudian author John melakukan *clone* dengan perintah `$ git clone git@git.gitserver:trafficLight.git`



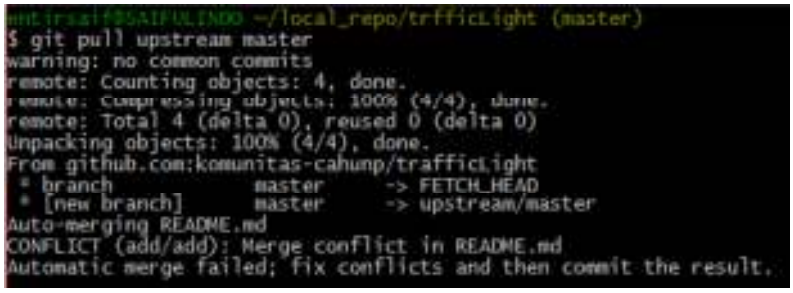
Gambar 3.28. ***Git Clone Shell***

Tahapan selanjutanya menambahkan *remote alias upstream*, ini berlaku pada *author* admincahunp dan John yaitu dengan perintah `$ git remote add upstream git@github.com:komunias-cahunp/trafficLight.git`, sebelum melakukan *push* maka harus menyiapkan *repository bare* pada github.com, sebagai contoh *author* John melakukan *push* ke *remote alias upstream* karena telah melakukan modifikasi kemudian *author* admincahunp melakukan *pull*.

Gambar 3.29. ***Git Remote Add Shell*** **– admincahunp dan John**



Gambar 3.30. ***Git Push Upstream Master*** **– John**



Gambar 3.31. *Git Pull Upstream Master* – admincahunp

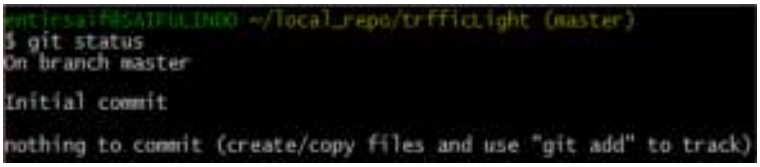
Kemudian bagaimana dengan *author* Aliece, ia bisa melakukan *clone* ke `git.gitserver` mapun ke github.com hanya saja *git server* mana yang Aliece melakukan *clone* pertama kali, sehingga *remote alias origin* mengarah kepada alamat *url* ketika melakukan *clone* sedangkan *remote alias upstream* cukup di tambahkan ke *git server* yang lainnya dengan perintah `$ git remote add upstream <url>`.

Gambar 3.32. *Git Clone Pada GitHub* – Aliece

b. Tampilan *Output*

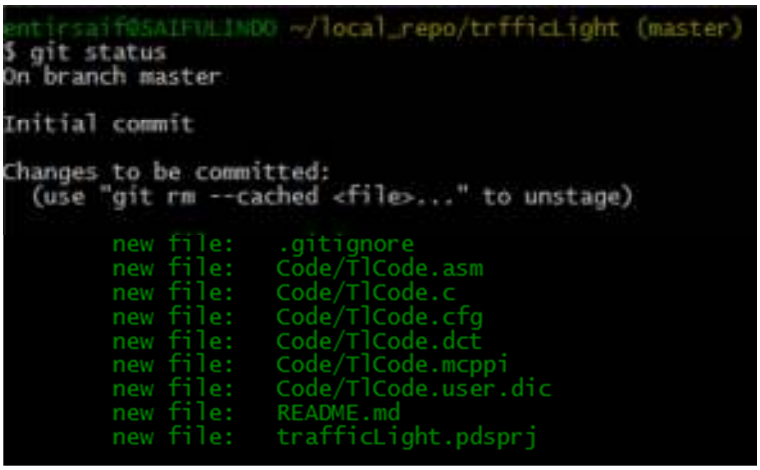
Pertama kali mengawali *git repository* dengan perintah `$ git init` maka, status *repository* yaitu berada pada cabang *master*, kemudian pengguna diperintahkan untuk membuat atau mengkopi *files* proyek untuk kemudian melakukan perintah `$ git add` untuk

tracked files selanjutnya melakukan perintah `$ git commit` untuk merekam perubahan *files project*.



Gambar 3.33. ***Git Status Init Shell***

Setelah melakukan `$ git add` maka status *repository* bahwa *files* yang dibuat atau di-kopi masuk *staging index* yang selanjutnya tinggal melakukan perintah `$ git commit`.



Gambar 3.34. ***Git Status Add Shell***

Setelah melakukan perintah `$ git commit` maka status *repository* yaitu *files* proyek sudah ter-rekam pada *database repository* sehingga tidak perlu melakukan perintah `$ git commit` karena *working directory* pada kondisi bersih dari *files* yang dimodifikasi, sehingga terjadi modifikasi *files, staging index, commit*

files dan seterusnya berulang sampai benar-benar proyek sesuai yang diinginkan.

Gambar 3.35. ***Git Status Commit Shell***

Selanjutnya ketika proyek sudah selesai melakukan tahapan *commit* maka sistem dapat melakukan *log* untuk mengetahui pesan perubahan apa yang di lakukan oleh *authors project* dan kapan *authors project* melakukan perubahan, ini yang menjadi *source code* ter-*manage* dengan baik.

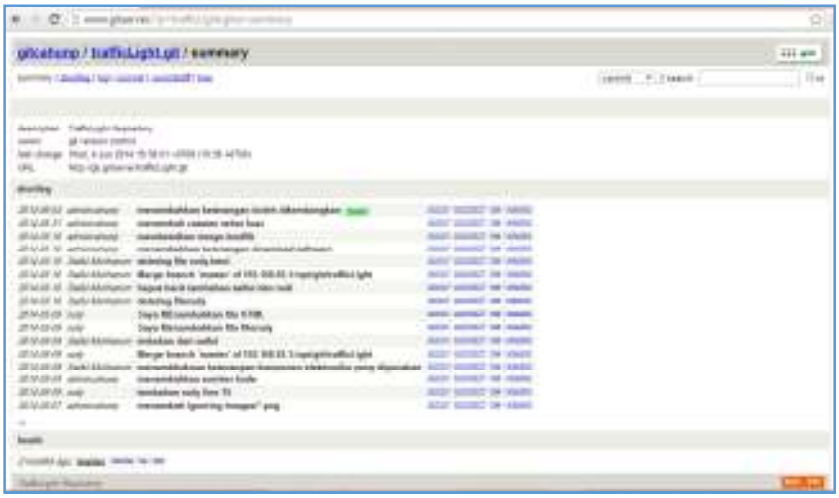
Gambar 3.36. ***Git Log Shell***

Setalah melakukan *commit* maka sistem akan memperbaharui *database* dengan nama kombinasi huruf dan angka dengan menggunakan konsep SHA pada *subderektory* `<.git/>` seperti : `<.git/objects/08/d3e0aec76e5cdf7c47a449563d082f5db0c79b>` Jadi, semakin banyak melakukan commit maka akan banyak pula databese baru pada *subdireckory* `<.git/object/.>`

Selanjutnya melakukan *remote alias* untuk melakukan kolaborasi, untuk menampilkannya dengan perintah `$ git remote -v` , sehingga muncul *remote alias* dengan nama *origin* pada *url* `git.gitserver`.

Gambar 5.37. *Git Remote List Shell*

Tahapan melakukan *push*, maka pada `git.gitserver` dapat di tampilkan dengan *tools git-web* dengan melakukan akses *url* `http://www.gitserver/`



Gambar 3.38. Antarmuka *Gitweb*

Tahapan selanjutnya *author* lain bisa melakukan *clone* misalnya *author* John melakukan *clone* dengan perintah `$ git clone git@git.gitserver:trafficLight.git` maka, secara

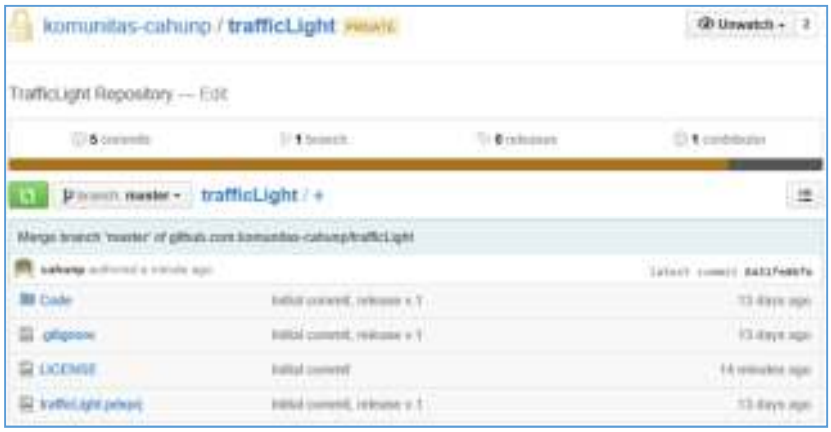
otomatis *remote alias* pada *repository*-nya John adalah *origin* dengan *url* `git@git.gitserver:trafficLight.git` , bisa ditampilkan dengan perintah `$ git remote –v`

Gambar 3.39. *Git Remote List* – John

Tehapan selanjutnya menambahkan *remote alias upstream* ini berlaku bagi *author* admincahunp maupun John, yaitu dengan perintah `$ git remote add upstream git@github.com:komunitas-cahunp/trafficLight.git` kemudian *remote alias* keduanya dapat ditempilkan dengan perintah `$ git remote –v .`

Gambar 3.40. *Git Remote List* – admincahunp dan John

Kemudian ketika *author* John melakukan *push* ke *remote alias upstream* karena telah melakukan modifikasi *project*, dengan perintah `$ git push upstream master`, tetapi sebelumnya harus di siapkan *repository bare* pada github.com.



Gambar 3.41. ***Repository TrafficLight on GitHub***

c. Modul Program

Pada tahapan ini akan menguraikan simulasi dua author yang melakukan kolaborasi proyek kalkulator dengan bahasa pemrograman C++, dengan keterangan *author*-nya yaitu John dan Aliece. *Author* John yang pertama kali membuat proyek kalkulator dengan aritmatika penjumlahan dan pengurangan kemudian *author* Aliece ikut bergabung untuk mengembangkan kalkulator dengan menambahkan aritmatika pembagian dan perkalian.

Simulasi diuraikan menjadi beberapa poin langkah yaitu menajemen proyek, membuat proyek, kolaborasi, modifikasi, dan sinkronisasi.

1) Manajemen Proyek

a) `git.gitserver`

Author admincahup menyiapkan repository bare dengan alamat `git@git.gitserver:calccah.git` menggunakan perintah `$`

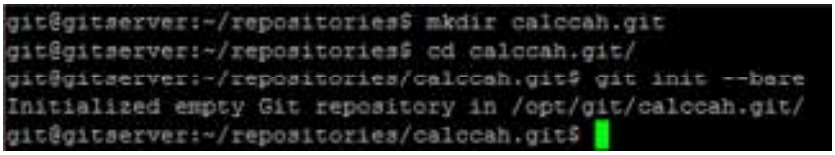
`git init --bare` dan membuka akses *repository* `calccah` pada *gitolite-admin*.

```
repo calccah
     RW+    = @all
     R      = daemon
```

Gambar 3.42. Menambahkan *Script Repo* `calccah`

author John dan Aliece mengirimkan `id_rsa.pub` dengan cara men-genarate `$ ssh-keygen -t rsa -C "alamat_email/nama_host"` ke *git server* melalui *author* admincahunp untuk diproses menggunakan *gitolite-admin* tepatnya di:

```
../gitolite-admin/keydir/*.pub.
                      ../aliece.pub
                      ../john.pub
```



Gambar 3.43. Membuat *Bare Repository*

b) github.com

Admincahunp juga menyiapkan *bare repository* pada github.com untuk social coding secara global dengan alamat

```
git@github.com:komunitas-cahunp:calccah.git
```

Gambar 3.44. *Url Repository* `calccah`

Dengan alamat *web* *github* `http://github.com/komunitas-cahunp/calccah.git`.

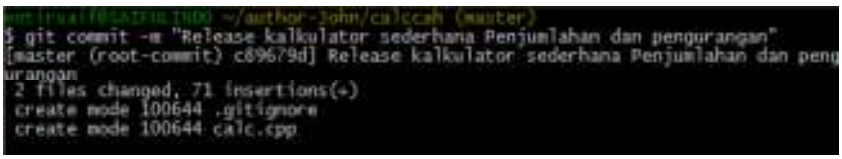
2) Membuat Proyek

Author John membuat proyek kalkulator pada computer local yang telah di *commit*. langkah – langkahnya :

```
$ mkdir calccah
$ cd calcah
$ git init
```

Membuat berkas `calccah.cpp` dan berkas `.gitignore` untuk membatasi extensi yang tidak perlu di *commit*.

```
$ git add .
$ git commit –m "Release kalkulator
sederhana Penjumlahan dan pengurangan"
```

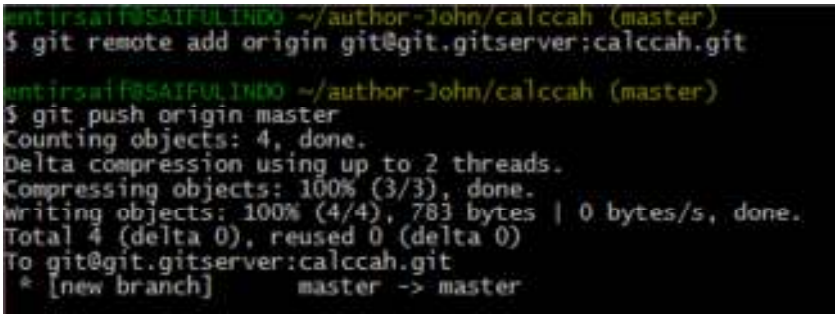


Gambar 3.45. *Git Commit* untuk *Release* Aplikasi Kalkulator

3) Kolaborasi

a) *Author* John

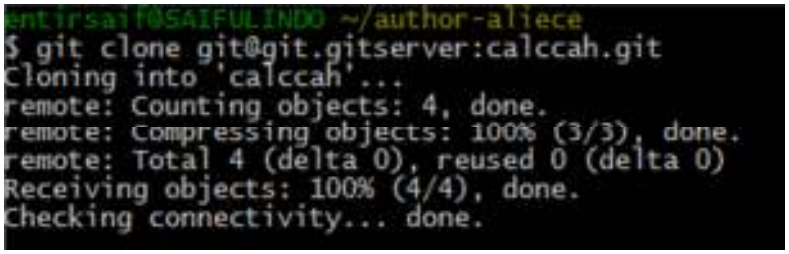
Menambah *remote alias* `# git remote add origin git@git.gitserver:calccah.git` selanjutnya `$ git push origin master`.



Gambar 3.46. Menambah *Remote Alias*

b) *Author* aliece

Melakukan *clone* `$ git clone \ git@git.gitserver:calccah.git`

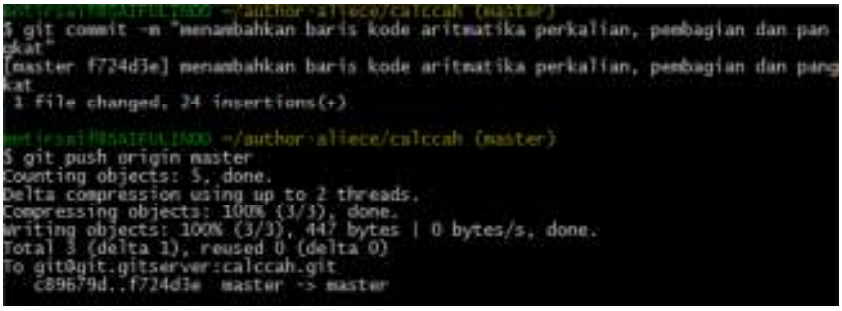


Gambar 3.47. *Author* Aliece Melakukan *Clone*

4) Modifikasi

Author aliece

Memodifikasi berkas `../caclcah/calc.cpp` menambahkan baris kode untuk aritmatika perkalian, pembagian dan pangkat, untuk berikutnya *pushing* ke `git.gitsrver`.

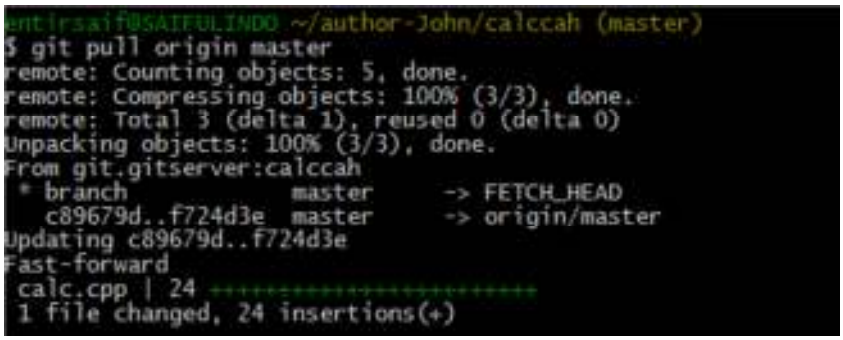


Gambar 3.48. Memodifikasi Berkas `calc.c`

5) Sinkronisasi dengan *github*

Author John

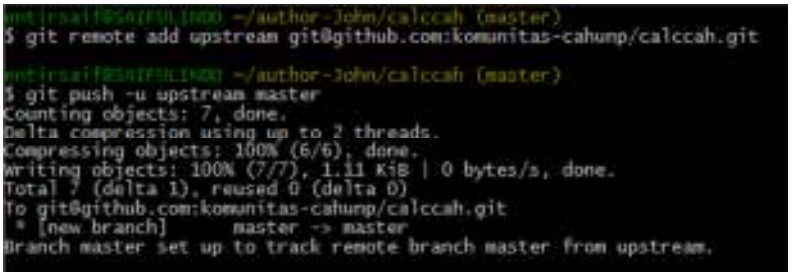
Update proyek `calccah.git` dengan author aliece maka john *pulling* menggunkan perintah `$ git pull origin master`.



Gambar 3.49. Singkronisasi dengan *pulling*

Berikutnya john menambahkan *remote alias upstream* dan selanjutnya *pushing.*

```
$ git remote add upstream \
  git@github.com:komunitas-cahunp/calccah.git
$ git push –u upstream master
```



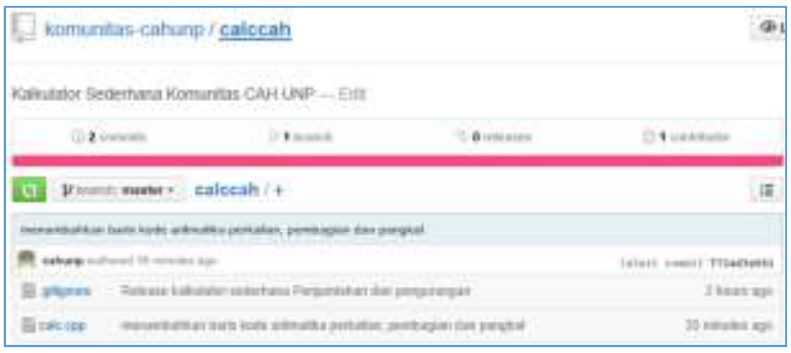
Gambar 3.50. Menambahkan *Remote Alias Upstream*

6) Hasil *Bare Repository*

Bare repository pada `git.gitserver`

Gambar 3.51. *Bare Repository* pada `git.gitserver`

Bare Repsitory pada github.com



Gambar 3.52. *Bare Repositroy* pada github.com

C. Lokasi dan Subjek Peneletian

1. Nama Lokasi

   Nama lokasi penilitian yaitu Fakultas Teknik Universitas Nusantara PGRI Kediri, Alamat Jl. KH. Achmad Dahlan 76 Kediri, Gedung L – lan - tai 3, Telp. 0354 – 770202.

2. Subyek Penelitian

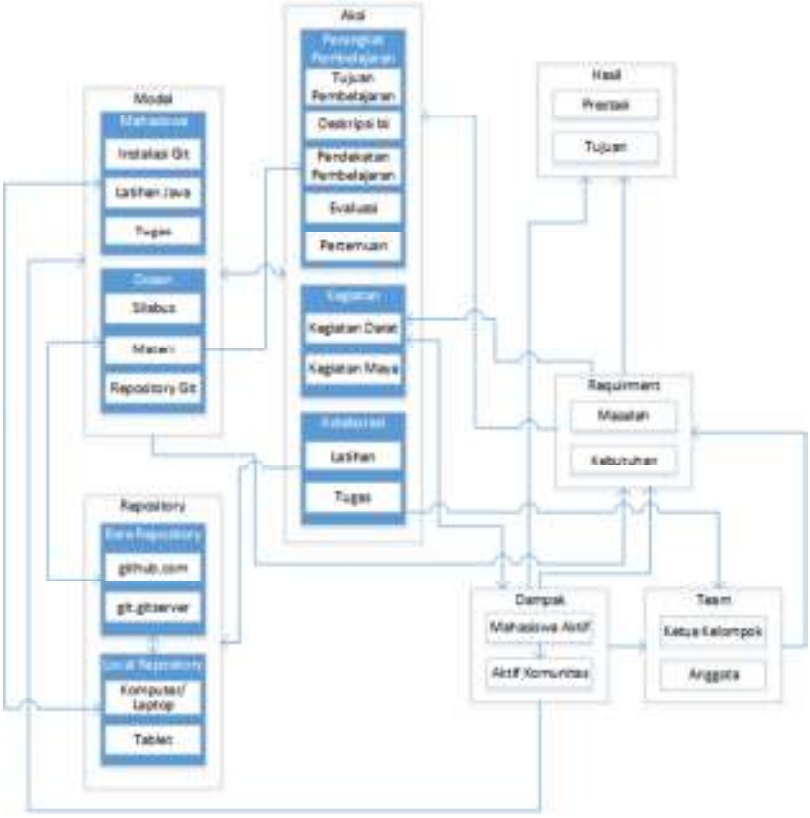
   Subyek Penelitian adalah mahasiswa Fakultas Teknik jurusan Teknik Informatikan tingkat II pada mata kuliah pemrograman java.

D. Uji Coba Model

1. Desain Uji Coba

   Diagram model kolaborasi social coding dikembangkan dari Diagram Model Motivasi Komunitas yang dirumuskan oleh Romi Satria Wahono (2007: 5), yang kemudian di rumuskan oleh peneliti sesuai dengan kondisi seperti berikut :

   Diagram yang berwarna menunjukkan segmen yang dilaksanakan dalam menerapkan *Git Server* sebagai media pembelajaran, sedangkan diagram yang tidak berwarna menunjukkan segmen yang muncul dari sebab dan akibat yang mempengaruhi maupun dipengaruhi.



Gambar 3.53. Diagram Model Pembelajaran *Social Coding*

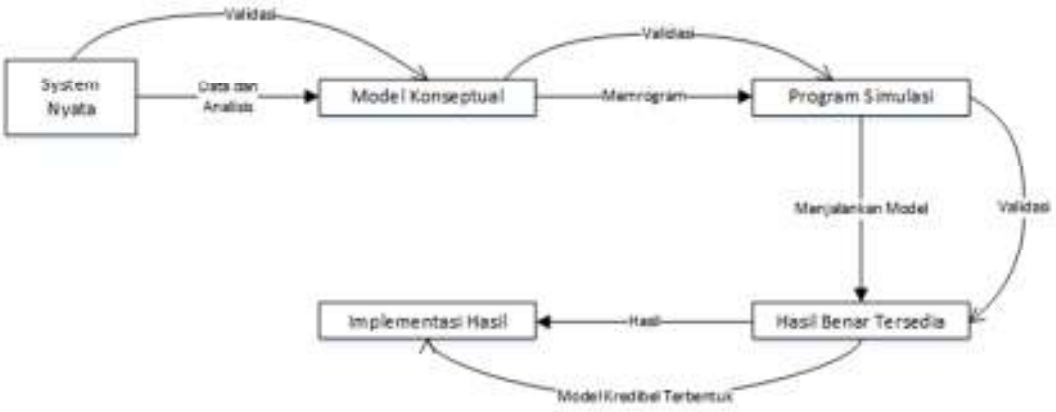
2. Subjek Uji Coba

Subyek uji coba berjumlah 40 mahasiswa Fakultas Teknik jurusan Teknik Informatika tingkat II kelas dua-a tahun akademik 2014/2015 terdiri dari 23 mahasiswa laki – laki dan 17 mahaisiswa perempuan. Obyek yang dijadikan pembahasan adalah mata kuliah pemrograman java. Pemilihan mata kuliah dengan mempertimbangkan bahwa materi tersebut dapat memberi tantangan menarik bagi mahasiswa.

E. Validasi Model

Menurut Hoover dan Perry validasi adalah proses penentuan apakah

model, sebagai konseptualisasi atau abstraksi, merupakan representasi berarti dan akurat dari sistem nyata. (Plonka dan Olling, 2013:343), sedangkan menurut Law dan Kelton validasi adalah penentuan apakah model konseptual simulasi adalah representasi akurat dari sistem nyata yang sedang dimodelkan (Plonka dan Olling, 2013:343).



Gambar 3. 54. Relasi verifikasi, validasi dan Pembentukan Model.

Ketika membangun model simulasi sistem nyata, harus melewati beberapa tahapan atau level permodelan. Pertama, membangun model konseptual yang memuat elemen sistem nyata. Dari model konseptual ini kemudian membangun model logika yang memuat relasi logis antara elemen sistem juga variabel eksogenus yang mempengaruhi sistem. Menggunakan model logika ini, lalu dikembangkan program komputer.

F. Instrumen Pengumpulan Data

Pengembangan dan validasi instrument di uraikan oleh Aimey (2012:homepage) seperti berikut :

1. Pengembangan Instrumen

Dalam pengembangan instrumen ini akan menggunkan beberapa teknik pengumpulan data yaitu sebagai berikut :

a) Kuesioner (Angket)

Kuesioner (Angkat) merupakan teknik pengumpulan data yang dilakukan dengan cara memberikan seperangkat pertanyaan atau pernyataan tertulis kepada responden untuk dijawabnya, dimana tidak langsung bertanya jawab dengan responden. Dalam penyusunan angket perlu diperhatikan beberapa hal : Pertama, sebelum butir – butir pertanyaan atau pernyataan ada pengantar atau petunjuk pengisian. Kedua, butir – butir pertanyaan dirumuskan secara jelas menggunakan kata – kata yang lazim digunakan (populer), kalimat tidak terlalu panjang. Ketiga, untuk setiap pertanyaan atau pernyataan terbuka dan tersetruktur disediakan kolom untuk menuliskan jawaban atau respon dari responden secukupnya.

b) Dokumentasi

Dokumentasi merupakan suatu teknik pengumpulan data dengan menghimpun dan menganalisis dokumen – dokumen, baik dokumen tertulis, gambar, maupun elektronik. Dokumen – dokumen yang dihimpun dipilih yang sesuai dengan tujuan dan fokus masalah. Metode dokumentasi dapat dilaksanakan dengan dua cara, yaitu : Pertama, pedoman dokumentasi yang memuat garis – garis besar atau

kategori yang akan dicari datanya. Kedua, Check-list yaitu daftar variable yang akan dikumpulkan datanya.

2. Validasi Instrumen

a) Validasi Isi

Untuk melakukan validasi isi dilakukan analisis pada masing – masing butir dan keseluruhan butir, untuk masing – masing butir menggunakan formula Shultz dan Whitnay (2005:89), sedangkan untuk keseluruhan butir menggunkan formula Gregory. Analisis masing – masing butir dan keseluruhan butir diuraikan seperti berikut:

(1) Analisis masing – masing butir.

Digunakan formula : CVR = $\frac{n1-N/2}{N/2}$ Dimana : n1 adalah banyanya penelaah yang menyatakan essensial, N adalah banyaknya penelaah dan, validitas isi dikatakan memenuhi syarat jika CVR ≥ 0,75.

(2) Analisis seluruh butir.

Diguakan formula : CV = $\frac{D}{A+B+C+D}$ Diamana : A adalah jumlah item yang kurang relevan kedua panelis, B adalah jumlah item yang kurang relevan menurut panelis I dan relevan menurut panelis II, C adalah jumlah item relevan menurut panelis I dan yang kurang relevan menurut panelis II dan, D adalah jumlah item yang relevan menurut kedua panelis.

b) Validasi Kriteria

Prosedur pendekatan validitas berdasarkan kriteria menghendaki

tersedianya kriteria ekternal yang dapat dijadikan dasar pengujian skor tes. Suatu kriteria adalah variabel perilaku yang akan diprediksi oleh skor tes atau berupa suatu ukuran lain yang relevan. Untuk melihat tingginya validitas berdasarkan kriteria dilakukan komputasi korelasi product moment antara skor tes (X) dengan skor kriteria (Y):

$$r_{xy} = \frac{\{N \sum XY\} - \{X\}\{Y\}}{\sqrt{\{N \sum X^2 - (\sum X)^2\}\{N \sum Y^2 (Y)^2\}}}$$

Keterangan :

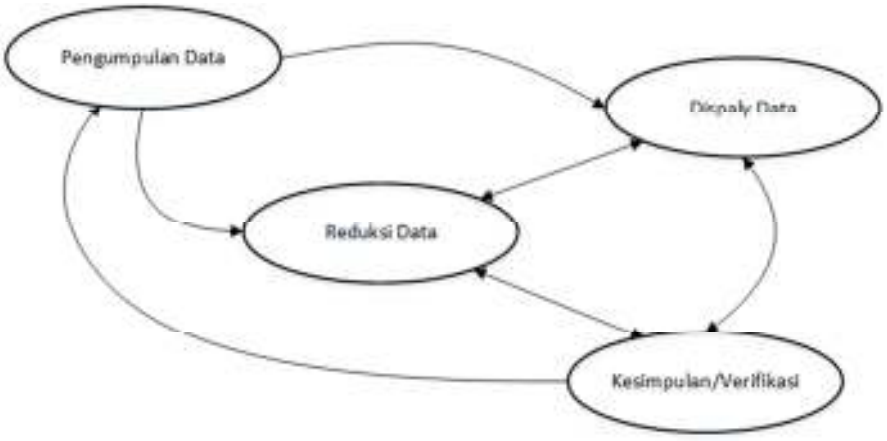
r_{xy} = Koefisien korelasi antara X dan Y X= Skor butir angket

N = Jumlah Subyek uji coba Y= Skor Total Angket

G. Teknik Analisis Data

Teknik analisis data telah dijelaskan oleh Ali Sya,ban (2005:69) seperti berikut ini:

1. Tahapan – tahapan Analisis Data



Gambar 3.55. Komponen – Komponen Analisis Data Model Interaktif Miles dan Huberman (1992:12)

2. Norma Pengujian

Norma pengujian berdasarkan komponen–komponen analisis data model interaktif telah dirumuskan oleh Miles dan Huberman (1992:12) sebagai berikut:

a) Pengumpulan Data

Pengumpulan data berupa data–data mentah dari hasil Penelitian, yaitu hasil angket dan dokumentasi.

b) Reduksi Data

Setelah data terkumpul dari hasil angket dan dokumntasi serta bahan–bahan lain yang ditemukan dilapangan dikumpulkan dan diklarifikasi dengan membuat catatan–catatan ringkasan, mengkode untuk menyesuaikan menurut hasil Penelitian.

c) Penyajian Data

Data yang sudah dikelompokkan dan sudah disesuaikan dengan kode – kodenya, kemudian disajikan dalam bentuk tulisan deskriptif agar mudah dipahami secara keseluruhan dan juga dapat menarik kesimpulan untuk melakukan penganalisisan dan Penelitian selanjutnya.

d) Kesimpulan atau Verifikasi

Hasil Penelitian yang telah terkumpul dan terangkum harus diulang kembali dengan mencocokkan pada reduksi data dan *display* data, agar kesimpulan yang telah dikaji dapat disepakati untuk ditulis sebagai laporan yang memiliki tingkat kepercayaan yang benar.

BAB IV

DESKRIPSI, INTERPRETASI DAN PEMBAHASAN

A. Hasil Studi Pendahuluan

1. Deskripsi Hasil Studi Lapangan

   Peneliti menemukan bahwa karakter belajar mahasiswa dalam mengerjakan tugas dan latihan mata kuliah pemrograman masih mempersoalkan masalahnya yang lokal, media belajar pemrograman yang masih individu, serta kurangnya mahasiswa bersikap *social coding* dalam menyelesaikan permasalahan pemrogramannya.

   Kemudian mahasiswa yang tergabung dalam KCU baik itu sebagai anggota maupun pengurus, dalam kegiatan komunitas peneliti menemukan bahwa sebagian besar mereka masih pasif, maksudnya keikutsertaan mereka dalam komunitas untuk mendapatkan materi, bukan untuk berbagi ataupun memberi materi.

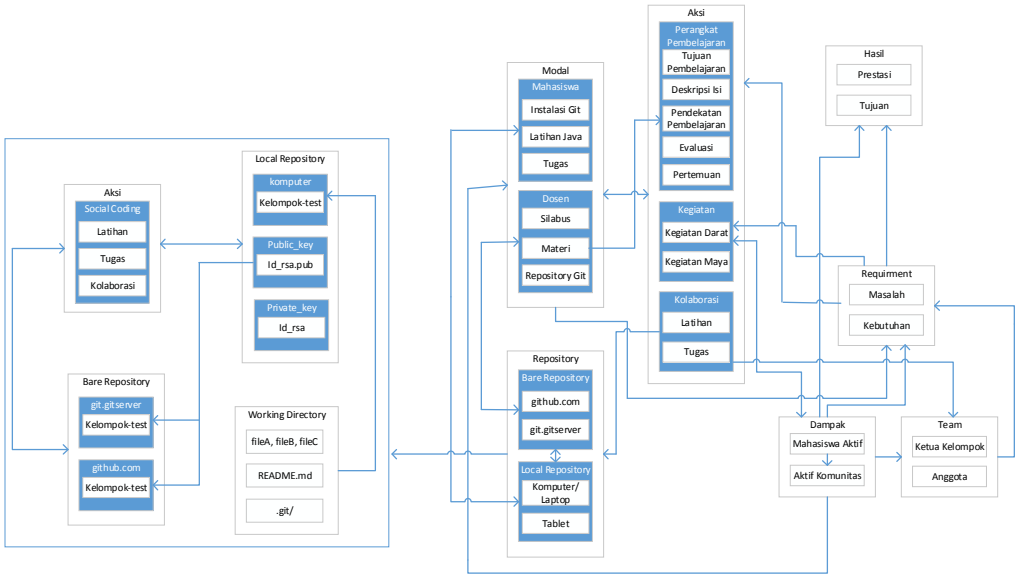
2. Interpretasi Hasil Studi Lapangan

   Peneliti menilai bahwa keaktifan anggota ataupun pengurus komunitas erat kaitannya terhadap bagaimana karakter belajar mahasiswa, sehingga perlu adanya sebuah program pembelajaran yang dapat meningkatkan motivasi belajar mahasiswa. Seperti mempersoalkan permasalahan pemrograman dengan lebih luas, mengerjakan tugas dan latihan pemrograman yang terdistribusi, serta belajar pemrograman

dengan konsep *social coding,* dengan demikian mahasiswa dapat mempercepat proses berfikir tentang pemrograman dan bahkan sebagai bentuk mahasiswa dapat merefleksikan hasil belajar mereka dikelas.

3. Desain Awal (draft) Model

Desain Awal ini adalah diambil dari desain uji coba yang telah di gambarkan dengan menambahkan penjabaran pada elemen *Repository* yang kemudian digunakan untuk uji kelayakan pada uji coba terbatas pertama.



Gambar 4.1. Diagram Model Pembelajaran *Social Coding* dengan Diagram Uji Kelayakan.

B. Pengujian Model Terbatas

1. Uji Validasi Ahli dan Praktisi

Dalam Uji Validasi peneliti membagi menjadi dua yaitu uji

kelayakan dan kemanfaatan, uji kelayakan yang dimaksud yaitu menilai apakah *Git Server* memiliki kelayakan untuk di operasikan, sedangkan uji kemanfaatan dilakukan untuk mendapatkan sebuah model rancangan akhir dan menilai sikap belajar mahasiswa ketika menggunakan *Git Server*.

2. Uji Coba Lapangan (Uji Coba Terbatas)

Dalam penelitian ini uji coba terbatas dilakukan sebanyak dua kali yaitu dengan rincian pengujian terbatas pertama dilakukan kepada tujuh mahasiswa dari jurusan teknik informatika dan sistem informasi, kedua dilakukan kepada sebelas mahasiswa dari jurusan teknik informatika, berikut rincian hasil uji coba terbatas :

a) Uji Coba Pertama

Hasil uji coba pertama pada uji kepahaman terukur 83,42% kategori sangat baik, sedangkan uji kelayakan terukur 87,79% kategori sangat layak.

Perbaikan hasil uji coba pertama yaitu pada sisi server dengan menambahkan nameserver `192.168.92.1` dan `192.168.91.1` serta mengkonfigurasi server sebagai *NAT*. Pada sisi client tindakan yang dilakukan adalah melakukan *disable firewall* (antivirus), *virtual adapter*, dan *Add-ons*.

b) Uji Coba Kedua

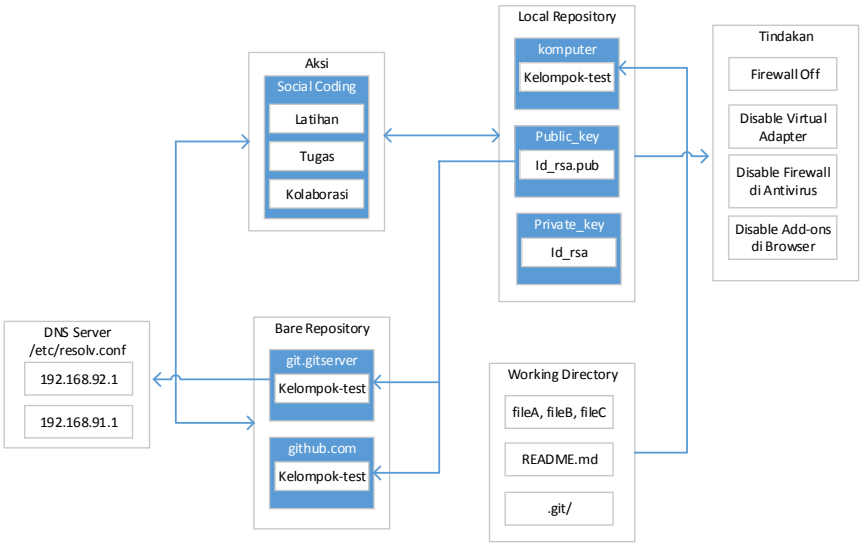
Hasil uji coba yang kedua pada uji kepahaman terukur 78,54% kategori baik, sedangkan uji kelayakan terukur 85,70% kategori

sangat layak. Meskipun hasil uji coba kedua kelayakan terukur turun dari uji coba pertama ini dinilai wajar karena uji kepahaman juga terukur turun, meskipun demikian uji kelayakan masih dalam kategori sangat layak.

Perbaikan hasil uji coba kedua yaitu membuatkan modul praktikum *Git Server Manual*, untuk memandu mahasiswa latihan mengoperasikan *Git Server*.

3. Desain Model Hasil Uji Coba Terbatas

Setelah dilakukan perbaikan berdsarkan uji terbatas pertama dan kedua maka didapatkan desain berikut :



Gambar 4.2. Desain Model Hasil Uji Coba Terbatas.

C. Pengujian Model Perluasan

1. Deskripsi Uji Coba Luas

Dalam uji coba luas dilakukan pengujian *Git Server* kepada

mahasiswa tingkat dua jurusan teknik informatika kelas dua-a dan dua-g. Dengan mensimulasikan *Git Server* dengan mata kuliah pemrograman.

Uji coba luas dimaksudkan untuk mendapatkan rancangan model akhir dan menilai kemanfaatan model untuk mengukur sikap belajar mahasiswa.

2. Refleksi dan Rekomendasi Hasil Uji Coba Luas

Kepahaman mahasiswa dalam mengoperasikan *Git Server* terukur kategori baik yaitu 64,80 % dengan rincian 59,60 % mahasiswa tidak bisa mengoperasikan *Git* karena alasan tertentu, 56,80 % mahasiswa bisa mengoperasikan *Git* tanpa bantuan, 74,20 % mahasiswa memiliki kemauan keras untuk bisa mengoperasikan git, 60,00 % mahasiswa dapat mengoperasikan *Git* ketika dikolaborasikan dengan *Git Server* tanpa bantuan, dan 73,30 % mahasiswa dapat belajar bersama pemrograman menggunakan *Git Server*.

Kelayakan model digambarkan dalam uji kemampuan pada butir soal kelima, sehingga kelayakan model terukur 73,30 % kategori layak, yaitu layak untuk dioperasikan sebagai media belajar mata kuliah pemrograman.

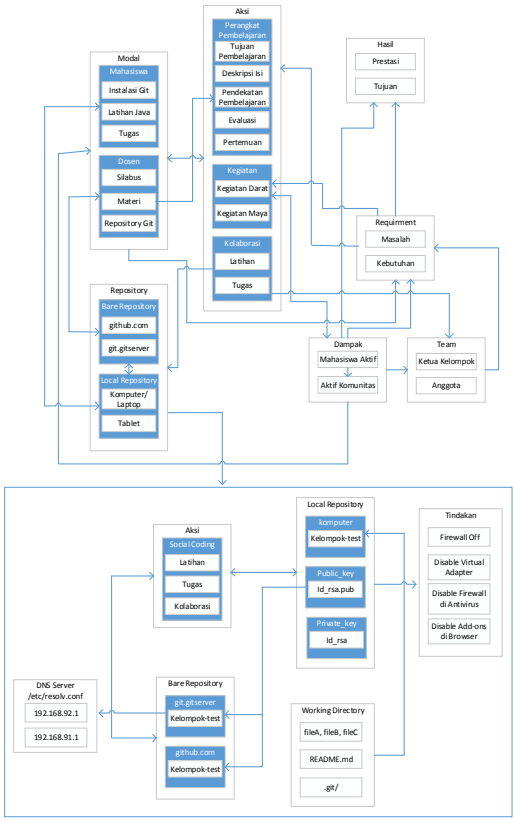
Kemanfaatan *Git Server* untuk mengukur sikap belajar mahasiswa terukur kategori baik yaitu 77,60 % dengan rincian 82,00 % mahasiswa setuju bahwa *Git Server* dapat digunakan sebagai media belajar bersama mata kuliah pemrograman, 66,53 % mahasiswa lebih aktif dalam pembelajaran mata kuliah pemrograman ketika menggunakan *Git Server*

sebagai media belajarnya, 80,00 % mahasiswa ingin mengasah kemampuan pemrogramannya dengan mahasiswa lain, yang sama – sama menggunkan *Git Server*, dan 81,63 % mahasiswa menginginkan belajar dan berbagi *source code* dengan konsep *social coding* menggunakan *Git Server*.

Perbaikan setelah melakukan uji coba luas adalah membuatkan panduan penggunaan *Git Server* untuk Instruktur atau Dosen.

3. Model Hipotetik

Model hipotetik didapatkan dari perbaikan uji coba terbatas dan luas yang telah dilakukan, sehingga didapatkan model seperti berikut :



Gambar 4.3. Diagram Model Pembelajaran *Social Coding* Hipotetik.

D. Validasi Model

1. Deskripsi Hasil Uji Validasi

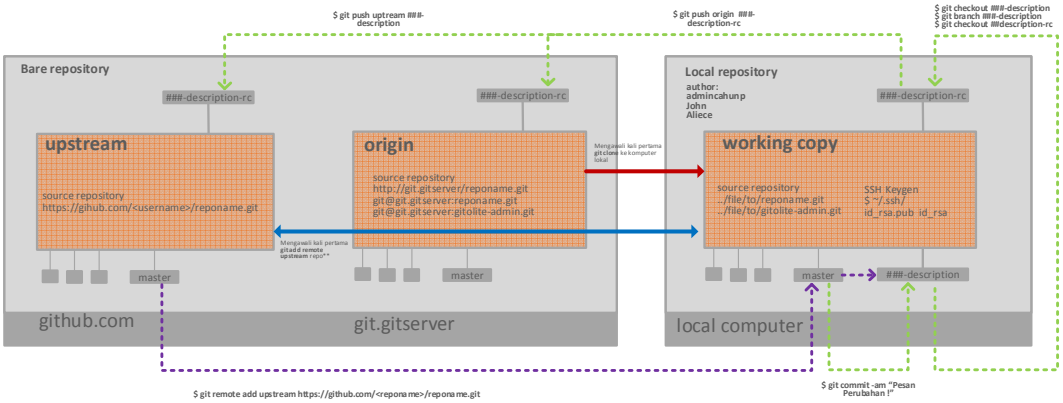
Dalam validasi model peneliti membagi menjadi tiga yaitu model konseptual, model logikal, dan model simulasi yang kemudian diujikan kepada ahli atau dosen yang kemudian disebut validator sebanyak lima validator, hasil uji validasi model konseptual terukur bahwa semua validator menyatakan bahwa model mengandung semua elemen, kejadian dan relasi yang sesuai, serta model dapat menjawab pertanyaan permodelan.

Hasil uji verifikasi dan validasi model logikal yang diawalai dengan uji verifikasi terukur bahwa semua validator menyatakan bahwa kejadian direpresentasikan dengan benar dan rumus relasi sudah benar, kemudian uji validasi terukur bahwa semua validator menyatakan bahwa model memuat semua kejadian yang ada pada model konseptual, serta model memuat semua relasi yang ada dalam model konseptual.

Hasil uji verifikasi dan validasi model simulasi yang diawali dengan uji verifikasi terukur bahwa semua validator meyatakan kode komputer sudah memuat semua aspek model logika, dan dua validator menyatakan model simulasi tidak mengandung kesalahan pengkodean berikutnya tiga validator menyatakan bahwa model simulasi mengandung kesalahan pengkodean, kemudian uji validasi terukur bahwa semua validator menyatakan bahwa model komputer merupakan representasi

valid dari sistem nyata dan model komputer dapat menduplikasi kinerja sistem nyata.

Perbaikan hasil uji validasi adalah menambahkan *source* `gitolite-admin.git` pada *bare repository origin* dan *local repository working copy*, ini perlu dicantumkan karena `gitolite-admin.git` sebagai manajemen *repository* dan *users* yang di kendalikan oleh *author* admincahunp, hasil perbaikan digambarkan seperti berikut :



Gambar 4.4. Model Simulasi Komputer

2. Interpretasi Hasil Uji Validasi

Berdasarkan hasil uji validasi menunjukkan bahwa rancangan model pembelajaran *social coding* dapat diterapkan pada mata kuliah pemrogaman, yaitu dapat digunakan untuk mengerjakan latihan dan tugas perkuliahan.

3. Kevalidan, Kepraktisan, dan Keefektifan Model

a) Kevalidan Model

Model dapat dinyatakan bahwa merupakan sebuah representasi valid dari sistem nyata serta model dapat menduplikasi

kinerja sistem nyata.

b) Kepraktisan Model

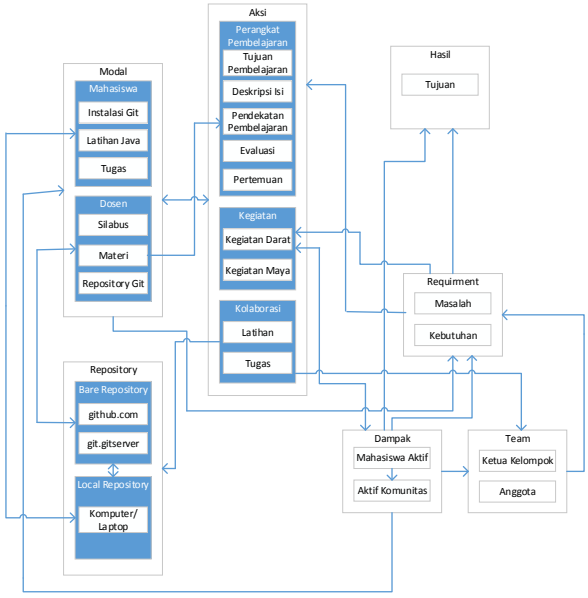
Model dinyatakan praktis karena dapat digunakan mengerjakan latihan dan tugas yang terdistribusi dalam satu tempat yaitu di *Git Server*.

c) Keefektifan Model

Model dinyatakan efektif karena pengumpulan latihan dan tugas dapat diketahui aktifitas mahasiswa ketika mengerjakan latihan dan tugas, dan tidak perlu menggunakan *via* lain untuk mengumpukan tugas.

4. Desain Akhir Model

Berdasarkan Uji coba dan validasi model maka diputuskanlah sebuah Diagram Model Pembelajaran *Social Coding,* seperti yang digambarkan berikut :



Gambar 4.5. Diagram Model Pembelajaran *Social Coding*.

E. Pembahasan Hasil Penelitian

1. Spesifikasi Model

   a. Proses

      Diawali dengan adanya proses pembelajaran dan perkuliahan kemudian adanya kebutuhan mengerjakan tugas dan latihan perkulian dan adanya masalah mengerjakan tugas dan latihan. Yaitu sebelumnya telah dipersiapkan *Git Server*-nya, kemudian sebelum memulai pembelajaran, perlu adanya latihan mengoperasikan *Git*, yang selanjutnya dilaksanakanlah proses kegiatan pembelajaran.

   b. Cara

      Cara mengoperasikan model perlu didukung dengan penguasaan materi tentang mengoperasi *version control Git*, cara megoperasikan telah dilampirkan dalam penelitian ini.

   c. Rincian

      Agar terlaksannya model maka rincian yang harus dipenuhi adalah 1) Adanya Proses pembelajaran yaitu ada dosen dan mahasiswa, 2) Adanya *bare repository* yaitu seabagai *remote reposiotory*, 3) Adanya laptop seabgai *local repository*, 4) Adanya *repository* latihan dan tugas 5) Masing – masing laptop sudah terinstalasi *version control*.

2. Prinsip – prinsip, Keunggulan dan Kelemahan Model

   a. Prinsip

      Asasnya model ini bisa di laksanakan, jika keduanya yaitu

instruktur/dosen dan mahasiswa bisa mengoperasikan Git secara *local repository* maupun dikolaborasikan dengan *Git Server*.

b. Keunggulan

Keaktifan mahasiswa dalam mengerjakan tugas maupun latihan dapat di log secara *up-to-date*, mahasiswa dapat merefisi karya mereka kapanpun, serta sebagai bentuk menanamkan konsep *social coding* dalam memahami program dan *code* secara terbuka.

c. Kelemahan

Kelemahan model ini mengharuskan mahasiswa menggunakan laptop sendiri – sendiri untuk mendapatkan hasil yang optimal, dibutuhkan keahlian untuk mengoperasikan perintah dasar operasi *file* dan direktori seperti pada *linux* dan perintah *local repositori* dan kolaborasi di *Git*.

3. Faktor Pendukung dan Penghambat Implementasi Model

a. Faktor Pendukung

Keahlian mengoperasikan *version control* seperti *Git* sangat dibutuhkan ketika mahasiswa bekerja di bidang *Software Development*.

b. Faktor Penghambat

Perangkat untuk menyediakan maupun mengakses *bare repository* terbatas, sempitnya pemahaman tentang *open source* dan *social coding*, mempersoalkan *code* pemrograman yang masih lokal.

BAB V

SIMPULAN, IMPLIKASI DAN SARAN

A. Simpulan

Berdasarkan hasil penelitian yang telah di uraikan pada bab IV dapat disimpulkan sebagai berikut:

1. a. Untuk merancang *Git Server* dibutuhkan keahlian dalam bidang jaringan dan pemrograman.
   b. *Git Server* dirancang harus dapat digunakan sebagai *public repository* maupun *private repository*.
   c. *Git Server* dirancang harus dapat memanajemen *repository* dan *users*.
   d. *Git Server* dirancang harus dapat digunakan sebagai kolaborasi pembelajaran mata kuliah pemrograman yang terdistribusi.
2. Rancangan *Git Server* dengan pendekatan *Github Social Coding* diimplikasikan dapat meningkatkan pembelajaran mahasiswa.

B. Implikasi

Berdasarkan hasil temuan dapat dikemukakan implikasi penelitian sebagai berikut:

1. Rancangan *Git Server* dengan Pendekatan *GitHub Social Coding* dapat digunakan sebagai media belajar mata kulih pemrograman.
2. Rancangan *Git Server* dengan Pendekatan *GitHub Social Coding* dapat meningkatkan pembelajaran mata kuliah pemrograman. Peningkatan

pembelajaran mahasiswa terukur 66,53 % mahasiswa lebih aktif dalam pembelajaran, 80,00 % mahasiswa menginginkan mengasah kemampuan pemrogramannya dengan mahasiswa lain, 81,63 % mahasiswa ingin merefleksikan apa dan bagaimana mereka belajar.

C. Saran

Berdasarkan penelitian yang telah diuraikan, maka dapat dikemukakan saran – saran sebagai berikut:

1. Rancangan *Git Server* ini memiliki dua alamat *url* yang berbeda yaitu `http://www.gitserver` dan `http://git.gitserver`, sebagai pengembangan penelitian kedepannya jadikan alamat *url* menjadi satu alamat *url* saja.
2. Peneliti berharap kepada para ahli yang berkecimpung dalam bidang pendidikan, untuk dapat meneliti lebih lanjut rancangan *Git Server* ini agar dapat diterapkan sebagai media pembelajaran mata kuliah pemrograman.
3. Rancangan *Git Server* ini dapat dikembangkan dengan membuat *user interface* pada tahapan pendaftaran *public key* ketika akan mengakses *repository* melalui protokol *SSH*.
4. Peningkatan pembelajaran mahasiswa yang terukur dalam penelitian ini dapat digunakan sebagai acuan, untuk pengembangan penelitian kedepannya.

DAFTAR PUSTAKA

Anggie, Baratadinata. 2012. *Mengenal Git*. (online). tersedia: https://dl.dropboxusercontent.com/u/2977985/books/MengenalGit.pdf, diunduh 24 Januari 2013.

Ali, Sya,ban. 2005. *Teknik Analiss Data Penelitian*. Makalah disampaikan pada Pelatihan Metode penelitian di Laboratorium Komputer UHAMKA, Pasar Rebo, Jakarta Timur, 13 Desember 2005. Dalam database stiead, (online). tersedia : http://www.stiead.ac.id/index.php/direktori-khusus/doc_download/44-diktat-analisis-data, diunduh 12 Maret 2015.

Aimey. 2012. *Penyusunan Instrumen Penelitian.* (online). tersedia : http://secercahsinaruntukhariesok.blogspot.com/2012/05/penyusunan-instrumen-penelitian.html, diunduh 18 Maret 2015.

Contributor, Team. 2012. *Ubuntu Server Guide*. (online). tersedia : https://help.ubuntu.com/12.04/serverguide/serverguide.pdf, diunduh 31 Agustus 2013.

Direktorat Jendral Pendidikan Tinggi Kementerian Pendidikan dan Kebudayaan. 2012. *Undang – undang Nomor 12 Tahun 2012 Tentang Pendidikan Tinggi*. (online). tersedia : http://sipuu.setkab.go.id/PUUdoc/17624/UU0122012_Full.pdf, diunduh 10 November 2014.

George, M. Doss. 1999. *Tipe Server RED HAT Linux.* Terjemahan Imam, Musthaqim. Jakarta : PT. Elex Media Komputindo.

Jan, Brazdil. 2013. *Automatic Pull Request Integration*. Disertasi. (online). tersedia : http://is.muni.cz/th/374346/fi_b/bakalarka__1_.pdf, diunduh 20 Maret 2014.

Jon, Loeliger. 2009. *Version Control with Git*. United State of America : O'Reilly Media. O'Really, (online), tersedia : http://oreilly.com/, diunduh 27 Februari 2014.

Jon, Loeliger. dan Matthew, McCollough. 2012. *Version Control with Git 2nd Edition*. United State of America : O'Reilly Media. O'Really, (online). tersedia : http://oreilly.com/, diunduh 1 September 2014.

Klint, Finley. 2011. *Github Has Surpassed Sourceforge and Google Code in Popularity*. (online). tersedia : http://readwrite.com/2011/06/02/github-has-passed-sourceforge, diunduh 23 April 2014.

Kenneth S., Shultz dan David J., Whitney. 2005. *Measurement Theory in Action*. (online). tersedia: http://download.portalgaruda.org/article.php?article=121551&val=649, diunduh 18 Maret 2015.

Linda, Suskie. 2002. *Strategies to Improve Student Learning.* (online). tersedia : https://www.mc.uky.edu/healthsciences/docs/Strategies-for-Improving-Student-Learning.pdf, diunduh 10 November 2014.

Linus, Torvalds. 2007. *Re: Kernel SCM Saga.* Linux-kernel (Mailling list). (online). Tersedia : http://marc.info/?l=linux-kernel&m=111288700902396, diunduh 18 Januari 2013.

Miftah, Andriansyah. Tanpa Tahun. *Verifikasi dan Validasi Model Simulasi.* (online). tersedia : http://didi.staff.gunadarma.ac.id/Downloads/files/5040/VERIFIKASI+DAN+VALIDASI+MODEL+SIMULASI.pdf , diunduh 14 Maret 2015.

Matthew B., Miles dan A. Michael, Huberman. 1994. *Qualitative Data Analysis – Second Edision*. (online). tersedia: https://vivauniversity.files.wordpress.com/2013/11/milesandhuberman1994.pdf, diunduh 18 Maret 2015.

Plonka, Frank dan Olling, Gustav J. (Eds). 2013. *Computer Applications in Production and Engineering.* IFIP TC5 International Conference on Computer Applications in Production and Engineering (CAPE '97) 5–7 November 1997, Detroit, Michigan, USA. (online). tersedia : http://www.springer.com/gp/book/9780412821103#otherversion=9780387352916, diunduh 18 Maret 2015.

Ridwan, Fajar. 2014. *10 Version Control System yang Harus Kamu Kenal.* (online). tersedia : http://www.codepolitan.com/10-version-control-system-yang-harus-kamu-kenal, diunduh Januari 2015.

Rob, Sanheim. 2013. *Three Million Users.* (online). tersedia : https://github.com/blog/1382-three-million-users, diunduh 08 April 2015.

Romi, Satria, W. 2007. *Sistem e-Learning Berbasis Model Motivasi Komunitas.* Jurnal Teknodik, Vol 1 (No. 1) 2007.

Scott, Chacon. 2009. *Pro Git*. (online). tersedia : https://github.s3.amazonaws.com/media/progit.en.pdf, diunduh 16 Januari 2013.

Sumarno, Alim, M.Pd, 2011. *Model Pembelajaran Kooperatif*. (online).tersedia: http://blog.elearning.unesa.ac.id/alim-sumarno/model-pembelajaran-kooperatif, diunduh Januari 2014.

Sitaram, Chamatry. 2014. *Gitolite Essentials*. Birmingham - Mumbai : Pack Publishing.

Wlodzimierz, Gajda. 2013. *Git Recieps*. New York : Apress. (online). tersedia : https://github.com/kospiotr/kospiotr.github.io/blob/master/resources/books/Git%20Recipes%20-%20Wlodzimierz%20Gajda.pdf, diunduh 16 Oktober 2014.

LAMPIRAN

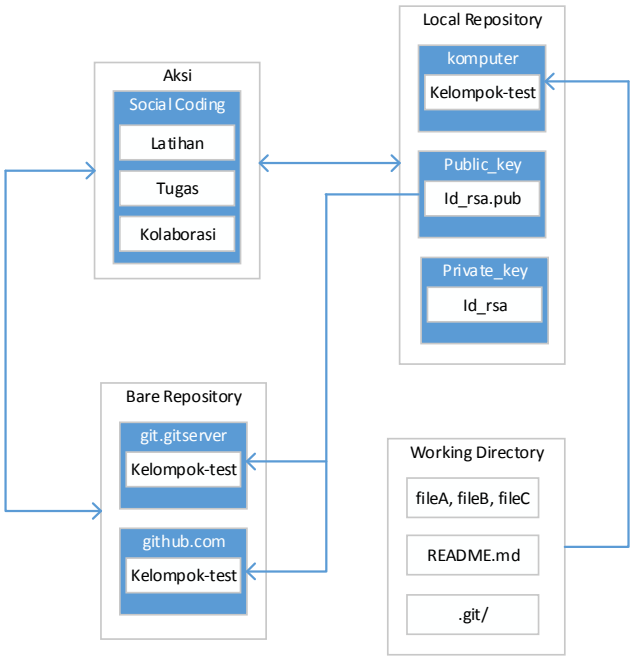
Lampiran 1. Instrumen Angket Partisipasi Uji Coba Terbatas Pertama.

INSTRUMEN ANGKET PARTISIPASI

A. **Petunjuk Umum**

Berkaitan dengan penyusunan skripsi tentang penelitian peningkatan pembelajaran mata kuliah pemrograman pada mahasiswa maka saya mohon partisipasi mahasiswa – mahasiswi sekalian untuk berpartisipasi memberikan respon untuk memberikan jawaban atas angket yang kalian terima. Pengisian angket ini tidak berpengaruh pada prestasi belajar kalian di kampus, oleh karena itu saya mohon untuk diisi sesuai dengan kondisi yang sebenarnya. Saya menjamin kerahasiaan identitas mahasiswa – mahasiswi sekalian. Atas perhatian diucapkan terima kasih.

B. Alur Kerja



C. **Petunjuk Khusus**

Berikan tanda checklist (√) pada kolom alternatif jawaban dari setiap pertanyaan maupun pernyataan pada kolom angket.

Keterangan :

Uji Kepahaman	Uji Kelayakan
SB = Sangat Baik B = Baik C = Cukup K = Kurang SK = Sangat Kurang	SL = Sangat Layak L = Layak C = Cukup K = Kurang SK = Sangat Kurang

D. **Angket Partisipasi Mahasiswa dalam Uji Kepahaman**

No.	Pertanyaan	Alternatif Jawaban				
		SB	B	C	K	SK
1.	Apakah perintah `git config` sudah bisa dipahami ?					
2.	Apakah perintah *git* seperti `git init`, `git add`, dan `git commit` sudah bisa dipahami ?					
3.	apakah perintah `ssh-keygen -t rsa -C "alamat-email/nama-host"` sudah bisa dipahami ?					
4.	Apakah perintah `git remote add upstream git@github.com:kelompok-test.git` sudah bisa dipahami ?					
5.	Apakah perintah *git* untuk kolaborasi seperti `git clone`, `git pull`, dan `git push` sudah bisa dipahami ?					

E. **Angket Partisipasi Mahasiswa dalam Uji Kelayakan**

No.	Pertanyaan	Alternatif Jawaban				
		SL	L	C	K	SK
1.	Apakah *SSID WiFi Access Point* Sudah sesuai ?					
2.	Apakah Ketika *workstation* Anda melakukan koneksi ke *Access Point* tidak mengalami masalah ?					
3.	Apakah proses mendapatkan *ip address workstation*, kinerjanya sudah baik ?					
4.	Apakah proses mendaftarkan `public_key` sudah sesuai ?					
5.	Apakah perintah git clone `git@git.gitserver:kelompok-test.git` kinerjanya sudah baik ?					
6.	Apakah perintah `git pull origin master` kinerjanya sudah baik ?					
7.	Apakah perintah `git push origin master` kinerjanya sudah baik ?					
8.	Apakah perintah git clone `http://git.gitserver/kelompok-test.git` kinerjanya sudah baik ?					
9.	Apakah alamat *url* `http://www.gitserver/` sudah sesuai ?					
10.	Apakah alamat *hostname* `git.gitserver` sudah sesuai ?					

11.	Apakah fitur *gitweb* kinerjanya sudah baik ?					

Pesan :

.....
.....
.....

Kediri,

Mahasiswa,

Sekala penilaian angket penelitian ini adalah :

1) Untuk pertanyaan positif :
   a) Skor 5 untuk pilihan jawaban SB dan SL
   b) Skor 4 untuk pilihan jawaban B dan L
   c) Skor 3 untuk pilihan jawaban C
   d) Skor 2 untuk pilihan jawaban K
   e) Skor 1 untuk pilihan awaban SK
2) Untuk pertanyan negatif :
   a) Skor 1 untuk pilihan jawaban SB dan SL
   b) Skor 2 untuk pilihan jawaban B dan L
   c) Skor 3 untuk pilihan jawaban C
   d) Skor 4 untuk pilihan jawaban K
   e) Skor 5 untuk pilihan awaban SK

Penentuan skor / nilai menggunkan rumus skala likert :

$$Rumus\ Index\ \% = \frac{Total\ Skor}{Y}\ x\ 100$$

$Y = Skor\ Tertinggi\ \ linkert\ x\ Jumlah\ Panelis$ $\quad T = Total\ Jumlah\ Panelis\ yang\ memilih$

$X = Skor\ Terendah\ Linkert\ x\ Jumlah\ Panelis$ $\quad Pn = Pilihan\ Angket\ Skor\ Likert$

Pertanyaan Positif : D.1, D.2, D.3, D.4, D.5, E.1, E.2, E.3, E.4, E.5, E.6, E.7, E.8, E.9, E.10, E.11

Pertanyaan Negatif : -

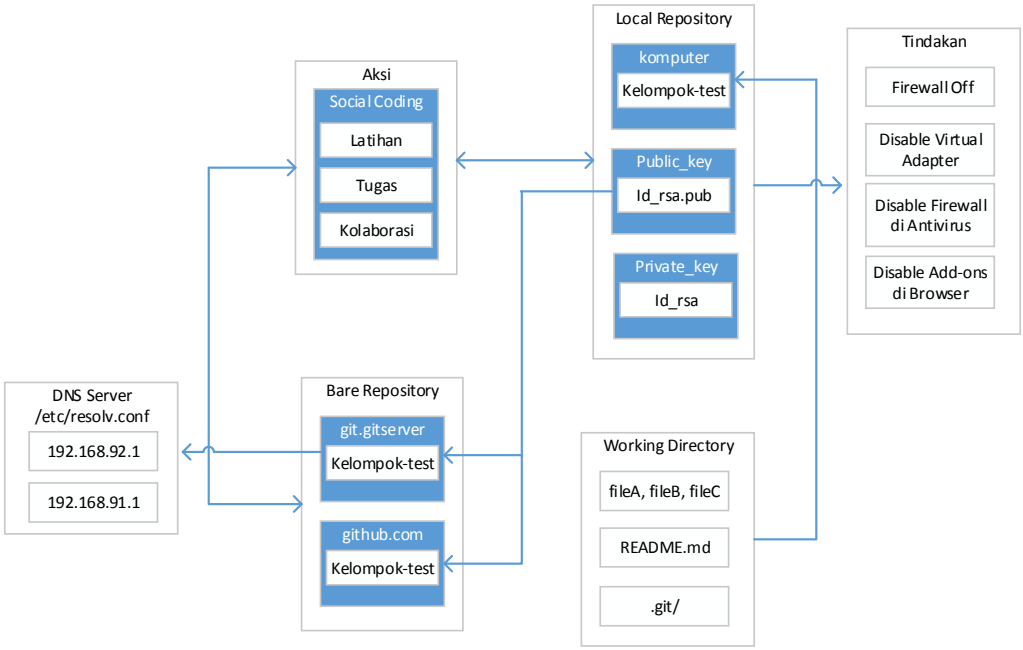
Lampiran 2. Instrumen Angket Partisipasi Uji Coba Terbatas Kedua.

INSTRUMEN ANGKET PARTISIPASI

A. Petunjuk Umum

Berkaitan dengan penyusunan skripsi tentang penelitian peningkatan pembelajaran mata kuliah pemrograman pada mahasiswa maka saya mohon partisipasi mahasiswa – mahasiswi sekalian untuk berpartisipasi memberikan respon untuk memberikan jawaban atas angket yang kalian terima. Pengisian angket ini tidak berpengaruh pada prestasi belajar kalian di kampus, oleh karena itu saya mohon untuk diisi sesuai dengan kondisi yang sebenarnya. Saya menjamin kerahasiaan identitas mahasiswa – mahasiswi sekalian. Atas perhatian diucapkan terima kasih.

B. Alur Kerja



C. Petunjuk Khusus

Berikan tanda checklist (√) pada kolom alternatif jawaban dari setiap pertanyaan maupun pernyataan pada kolom angket.

Keterangan :

Uji Kepahaman	Uji Kelayakan
SB = Sangat Baik B = Baik C = Cukup K = Kurang SK = Sangat Kurang	SL = Sangat Layak L = Layak C = Cukup K = Kurang SK = Sangat Kurang

D. **Angket Partisipasi Mahasiswa dalam Uji Kepahaman**

No.	Pertanyaan	Alternatif Jawaban				
		SB	B	C	K	SK
1.	Apakah perintah `git config` sudah bisa dipahami ?					
2.	Apakah perintah *git* seperti `git init`, `git add`, dan `git commit` sudah bisa dipahami ?					
3.	apakah perintah `ssh-keygen -t rsa -C "alamat-email/nama-host"` sudah bisa dipahami ?					
4.	Apakah perintah `git remote add upstream git@github.com:kelompok-test.git` sudah bisa dipahami ?					
5.	Apakah perintah *git* untuk kolaborasi seperti `git clone`, `git pull`, dan `git push` sudah bisa dipahami ?					

E. **Angket Partisipasi Mahasiswa dalam Uji Kelayakan**

No.	Pertanyaan	Alternatif Jawaban				
		SL	L	C	K	SK
1.	Apakah *SSID WiFi Access Point* Sudah sesuai ?					
2.	Apakah Ketika *workstation* Anda melakukan koneksi ke *Access Point* tidak mengalami masalah ?					
3.	Apakah proses mendapatkan *ip address workstation*, kinerjanya sudah baik ?					
4.	Apakah proses mendaftarkan `public_key` sudah sesuai ?					
5.	Apakah perintah git clone `git@git.gitserver:kelompok-test.git` kinerjanya sudah baik ?					
6.	Apakah perintah `git pull origin master` kinerjanya sudah baik ?					
7.	Apakah perintah `git push origin master` kinerjanya sudah baik ?					
8.	Apakah perintah git clone `http://git.gitserver/kelompok-test.git` kinerjanya sudah baik ?					
9.	Apakah alamat *url* `http://www.gitserver/` sudah sesuai ?					
10.	Apakah alamat *hostname* `git.gitserver` sudah sesuai ?					

11.	Apakah fitur *gitweb* kinerjanya sudah baik ?					

Pesan :

.....
.....
.....

Kediri,

Mahasiswa,

Sekala penilaian angket penelitian ini adalah :

1) Untuk pertanyaan positif :
   a) Skor 5 untuk pilihan jawaban SB dan SL
   b) Skor 4 untuk pilihan jawaban B dan L
   c) Skor 3 untuk pilihan jawaban C
   d) Skor 2 untuk pilihan jawaban K
   e) Skor 1 untuk pilihan awaban SK
2) Untuk pertanyan negatif :
   a) Skor 1 untuk pilihan jawaban SB dan SL
   b) Skor 2 untuk pilihan jawaban B dan L
   c) Skor 3 untuk pilihan jawaban C
   d) Skor 4 untuk pilihan jawaban K
   e) Skor 5 untuk pilihan awaban SK

Penentuan skor / nilai menggunkan rumus skala likert :

$$Rumus\ Index\ \% = \frac{Total\ Skor}{Y}\ x\ 100$$

$Y = Skor\ Tertinggi\ \ linkert\ x\ Jumlah\ Panelis$ $\quad T = Total\ Jumlah\ Panelis\ yang\ memilih$

$X = Skor\ Terendah\ Linkert\ x\ Jumlah\ Panelis$ $\quad Pn = Pilihan\ Angket\ Skor\ Likert$

Pertanyaan Positif : D.1, D.2, D.3, D.4, D.5, E.1, E.2, E.3, E.4, E.5, E.6, E.7, E.8, E.9, E.10, E.11

Pertanyaan Negatif : -

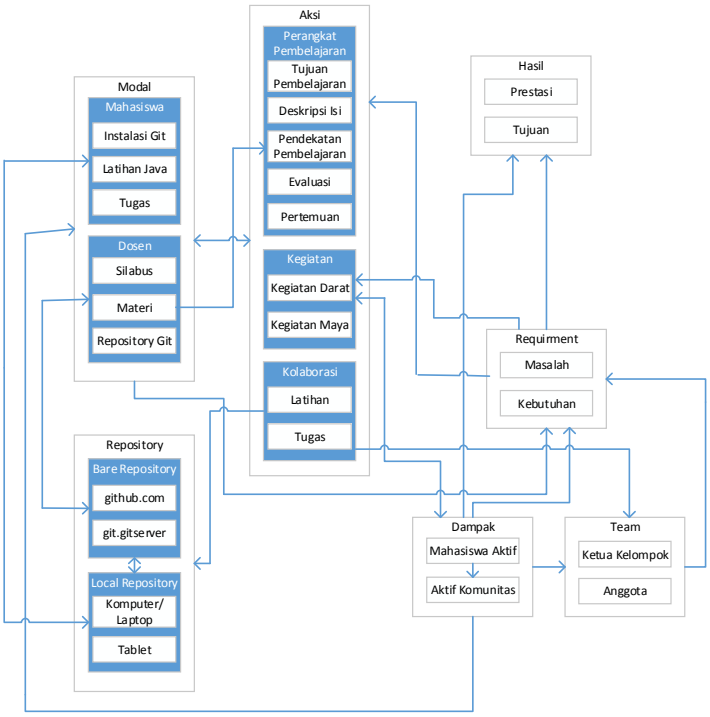
Lampiran 3. Instrumen Angket Partisipasi Uji Coba Luas.

INSTRUMEN ANGKET PARTISIPASI

A. Petunjuk Umum

Berkaitan dengan penyusunan skripsi tentang penelitian peningkatan pembelajaran mata kuliah pemrograman pada mahasiswa maka saya mohon partisipasi mahasiswa – mahasiswi sekalian untuk berpartisipasi memberikan respon untuk memberikan jawaban atas angket yang kalian terima. Pengisian angket ini tidak berpengaruh pada prestasi belajar kalian di kampus, oleh karena itu saya mohon untuk diisi sesuai dengan kondisi yang sebenarnya. Saya menjamin kerahasiaan identitas mahasiswa – mahasiswi sekalian. Atas perhatian diucapkan terima kasih.

B. Alur Kerja



C. Petunjuk Khusus

Berikan tanda checklist (√) pada kolom alternatif jawaban dari setiap pertanyaan maupun pernyataan pada kolom angket.

Keterangan :

Uji Kemampuan	Uji Kemanfaatan
SB = Sangat Baik/Benar B = Baik/Benar C = Cukup K = Kurang SK = Sangat Kurang	SS = Sangat Setuju/Sesuai S = Setuju/Sesuai C = Cukup K = Kurang SK = Sangat Kurang

D. **Angket Partisipasi Mahasiswa dalam Uji Kemampuan**

No.	Pertanyaan	Alternatif Jawaban				
		SB	B	C	K	SK
1.	Saya tidak bisa mengoperasikan *git* karena alasan tertentu.					
2.	Saya bisa mengoperasikan *git* tanpa bantuan.					
3.	Saya harus bisa mengoperasikan git sebisa mungkin bagaimanapun caranya.					
4.	Saya dapat mengoperasikan git ketika di kolaborasikan dengan *Git Server* tanpa bantuan.					
5.	Saya dapat belajar bersama pemrograman menggunakan *Git Server*.					

E. **Angket Partisipasi Mahasiswa dalam Uji Kemanfaatan**

No.	Pertanyaan	Alternatif Jawaban				
		SS	S	C	K	SK
1.	*Git Server* dapat digunakan sebagai media belajar bersama mata kuliah pemrograman.					
2.	Saya lebih aktif dalam pembelajaran mata kuliah pemrograman, ketika menggunakan *Git Server* sebagai media belajarnya .					
3.	Saya ingin mengasah kemampuan pemrograman saya dengan mahasiswa lain, yang sama - sama menggunakan *Git Server*.					
4.	Saya ingin belajar dan berbagi *source code* dengan konsep *social coding* menggunakan *Git Server*.					

Pesan :

.....

.....

.....

Kediri,

Mahasiswa,

Sekala penilaian angket penelitian ini adalah :

3) Untuk pertanyaan positif :
   f) Skor 5 untuk pilihan jawaban SB dan SL
   g) Skor 4 untuk pilihan jawaban B dan L
   h) Skor 3 untuk pilihan jawaban C
   i) Skor 2 untuk pilihan jawaban K
   j) Skor 1 untuk pilihan awaban SK
4) Untuk pertanyan negatif :
   f) Skor 1 untuk pilihan jawaban SB dan SL
   g) Skor 2 untuk pilihan jawaban B dan L
   h) Skor 3 untuk pilihan jawaban C
   i) Skor 4 untuk pilihan jawaban K
   j) Skor 5 untuk pilihan awaban SK

Penentuan skor / nilai menggunkan rumus skala likert :

$$Rumus\ Index\ \% = \frac{Total\ Skor}{Y}\ x\ 100$$

$Y = Skor\ Tertinggi\ \ linkert\ x\ Jumlah\ Panelis$ $\quad T = Total\ Jumlah\ Panelis\ yang\ memilih$

$X = Skor\ Terendah\ Linkert\ x\ Jumlah\ Panelis$ $\quad Pn = Pilihan\ Angket\ Skor\ Likert$

Pertanyaan Positif : D.2, D.3, D.4, D.5, E.1, E.2, E.3, E.4

Pertanyaan Negatif : D.1

Lampiran 4. Angket Validasi Model

Validasi Model

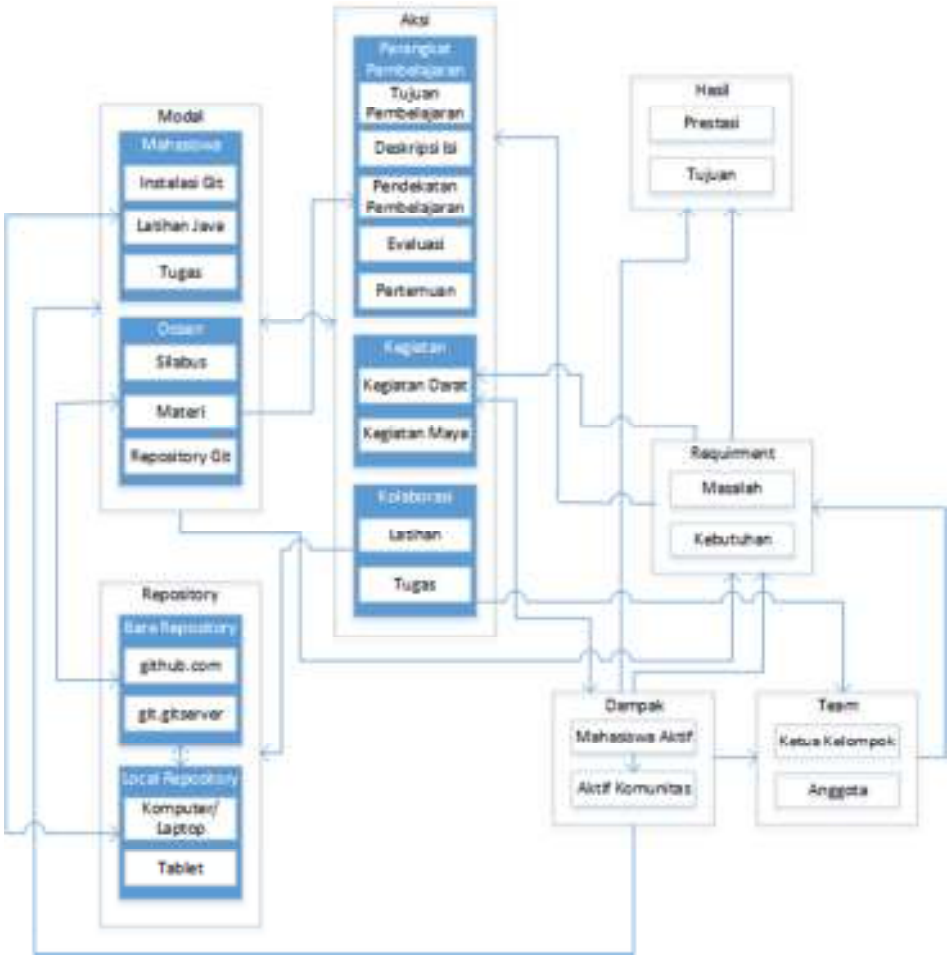
A. Petunjuk Umum

Berkaitan dengan penyusunan skripsi yang berjudul **Merancang Git Server dengan Pendekatan GitHub Social Coding dalam Peningkatan Pembelajaran Mahasiswa** maka saya mohon Bapak/Ibu Dosen sekalian untuk berpartisipasi memberikan respon atas angket validasi model ini.

B. Petunjuk Khusus

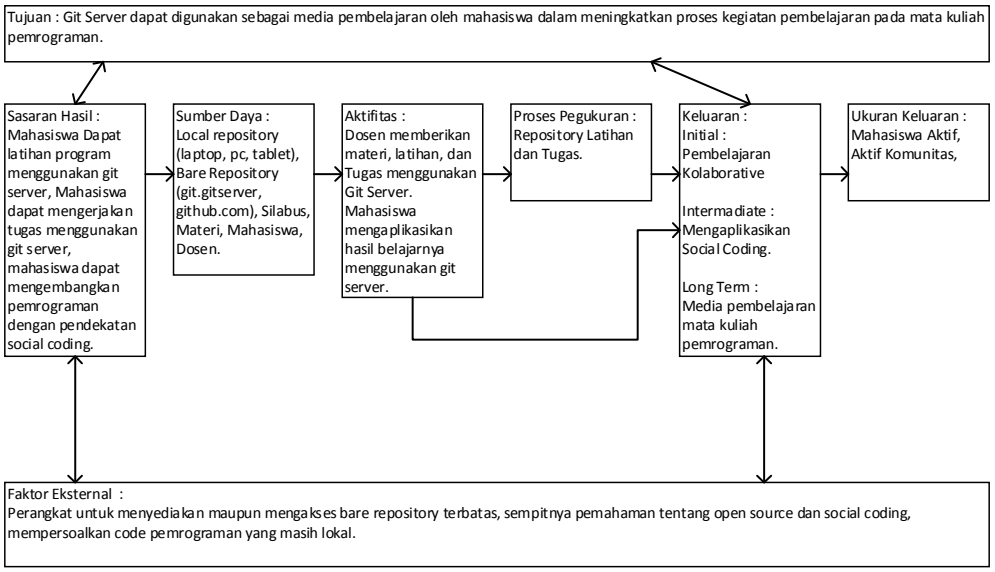
Mohon berikan tanda checklist (√) pada kolom alternatif jawaban dari setiap pertanyaan verifikasi dan validasi di bawah ini.

C. Angket Validasi Model



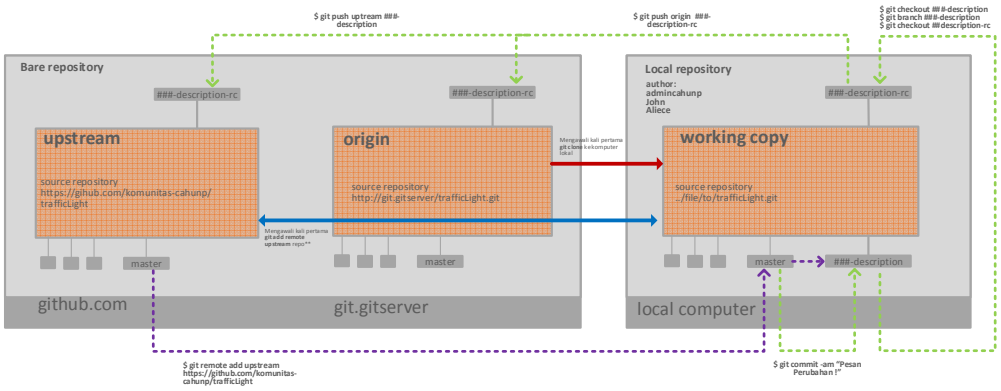
Model Konseptual

Verifikasi	Validasi	Alternatif Jawaban			
		Verifikasi		Validasi	
		Ya	Tidak	Ya	Tidak
	Apakah model mengandung semua elemen, kejadian dan relasi yang sesuai ?				
	Apakah Model dapat menjawab pertanyaan permodelan ?				



Model Logika

Verifikasi	Validasi	Alternatif Jawaban			
		Verifikasi		Validasi	
		Ya	Tidak	Ya	Tidak
Apkakah kejadian direpresentasikan dengan benar ?	Apakah model memuat semua kejadian yang ada pada model konseptual ?				
Apakah rumus relasi benar ?	Apakah model memuat semua relasi yang ada dalam model konseptual ?				



Model Simulasi

Verifikasi	Validasi	Alternatif Jawaban			
		Verifikasi		Validasi	
		Ya	Tidak	Ya	Tidak
Apakah kode komputer memuat semua aspek model logika ?	Apakah model komputer merupakan representasi valid dari sistem nyata ?				
Apakah model mengandung kesalahan pengkodean ?	Dapatkah model komputer menduplikasi kinerja sistem nyata ?				

Catatan :

.....
.....
.....
.....
.....

Kediri,
Validator,

Sekala penilaian angket penelitian ini adalah :

1) Untuk pertanyaan positif :
   a) Skor 2 untuk pilihan jawaban YA
   b) Skor 1 untuk pilihan jawaban TIDAK

2) Untuk pertanyan negatif :
   a) Skor 1 untuk pilihan jawaban YA
   b) Skor 2 untuk pilihan jawaban TIDAK

Penentuan skor / nilai menggunkan rumus skala likert :

$$Rumus\ Index\ \% = \frac{Total\ Skor}{Y}\ x\ 100$$

$Y = Skor\ Tertinggi\ \ linkert\ x\ Jumlah\ Panelis$ $\quad T = Total\ Jumlah\ Panelis\ yang\ memilih$

$X = Skor\ Terendah\ Linkert\ x\ Jumlah\ Panelis$ $\quad Pn = Pilihan\ Angket\ Skor\ Likert$

Lampiran 5. Hasil Uji Coba Terbatas Pertama.

Uji Coba Terbatas Pertama

A. Uji Kepahaman

Butir Pertama

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat baik	5	2	10
Baik	4	4	16
Cukup	3	1	3
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	29
		Y	35
		X	7
		Index%	82,86

Butir Kedua

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat baik	5	3	15
Baik	4	3	12
Cukup	3	1	3
Kurang	2	0	0
Sangat Kuran	1	0	0
		Total Skor	30
		Y	35
		X	7
		Index%	85,71

Butir Ketiga

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat baik	5	2	10
Baik	4	4	16
Cukup	3	1	3
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	29
		Y	35
		X	7
		Index%	82,86

Butir Keempat

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat baik	5	2	10
Baik	4	3	12
Cukup	3	2	6
Kurang	2	0	0
Sangat Kuran	1	0	0
		Total Skor	28
		Y	35
		X	7
		Index%	80

Butir Kelima

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat baik	5	3	15
Baik	4	3	12
Cukup	3	1	3
Kurang	2		0
Sangat Kurang	1		0
		Total Skor	30
		Y	35
		X	7
		Index%	85,71

$$Total\ Index\% = \frac{Total\ Persentase\ Seluruh\ Butir}{Jumlah\ Butir}$$

Total Index% = 83,4286 %

Kategori Sangat Baik

B. Uji Kelayakan

Butir Pertama

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	6	30
Layak	4	1	4
Cukup	3	0	0
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	34
		Y	35
		X	7
		Index%	97,14

Butir Kedua

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	3	15
Layak	4	2	8
Cukup	3	2	6
Kurang	2		0
Sangat Kuran	1		0
		Total Skor	29
		Y	35
		X	7
		Index%	82,86

Butir Ketiga

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	4	20
Layak	4	3	12
Cukup	3	0	0
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	32
		Y	35
		X	7
		Index%	91,43

Butir Keempat

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	1	5
Layak	4	4	16
Cukup	3	1	3
Kurang	2	0	0
Sangat Kuran	1	0	0
		Total Skor	24
		Y	30
		X	6
		Index%	80

Butir Kelima

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	3	15
Layak	4	2	8
Cukup	3	2	6
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	29
		Y	35
		X	7
		Index%	82,86

Butir Keenam

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	5	25
Layak	4	2	8
Cukup	3	0	0
Kurang	2	0	0
Sangat Kuran	1	0	0
		Total Skor	33
		Y	35
		X	7
		Index%	94,29

Butir Ketujuh

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	5	25
Layak	4	2	8
Cukup	3	0	0
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	33
		Y	35
		X	7
		Index%	94,29

Butir Kedelapan

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	3	15
Layak	4	3	12
Cukup	3	1	3
Kurang	2	0	0
Sangat Kuran	1	0	0
		Total Skor	30
		Y	35
		X	7
		Index%	85,71

Butir Kesembilan

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	1	5
Layak	4	5	20
Cukup	3	1	3
Kurang	2	1	2
Sangat Kurang	1	0	0
		Total Skor	30
		Y	35
		X	7
		Index%	85,71

Butir Kesepuluh

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	3	15
Layak	4	3	12
Cukup	3	0	0
Kurang	2	1	2
Sangat Kuran	1	0	0
		Total Skor	29
		Y	35
		X	7
		Index%	82,86

Butir Kesebelas

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	4	20
Layak	4	2	8
Cukup	3	1	3
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	31
		Y	35
		X	7
		Index%	88,57

$$Total\ Index\% = \frac{Total\ Persentase\ Seluruh\ Butir}{Jumlah\ Butir}$$

Total Index% = 87,7922 %

Kategori Sangat Layak

Lampiran 6. Hasil Uji Coba Tertbatas Kedua.

Uji Coba Tebatas Kedua

A. Uji Kepahaman

Butir Pertama

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Baik	5	2	10
Baik	4	9	36
Cukup	3	0	0
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	46
		Y	55
		X	11
		Index%	83,64

Butir Kedua

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Baik	5	2	10
Baik	4	7	28
Cukup	3	2	6
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	44
		Y	55
		X	11
		Index%	80

Butir Ketiga

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Baik	5	2	10
Baik	4	8	32
Cukup	3	0	0
Kurang	2	1	2
Sangat Kurang	1	0	0
		Total Skor	44
		Y	55
		X	11
		Index%	80

Butir Keempat

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Baik	5	0	0
Baik	4	7	28
Cukup	3	4	12
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	40
		Y	55
		X	11
		Index%	72,73

Butir Kelima

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Baik	5	3	15
Baik	4	4	16
Cukup	3	3	9
Kurang	2	1	2
Sangat Kurang	1	0	0
		Total Skor	42
		Y	55
		X	11
		Index%	76,36

$$Total\ Index\% = \frac{Total\ Persentase\ Seluruh\ Butir}{Jumlah\ Butir}$$

Total Index% = 78,54545 %

Kategori Baik

B. Uji Kelayakan

Butir Pertama

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	3	15
Layak	4	7	28
Cukup	3	0	0
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	43
		Y	50
		X	10
		Index%	86

Butir Kedua

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	2	10
Layak	4	6	24
Cukup	3	3	9
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	43
		Y	55
		X	11
		Index%	78,18

Butir Ketiga

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	2	10
Layak	4	9	36
Cukup	3	0	0
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	46
		Y	55
		X	11
		Index%	83,64

Butir Keempat

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	2	10
Layak	4	8	32
Cukup	3	0	0
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	42
		Y	50
		X	10
		Index%	84

Butir Kelima

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	6	30
Layak	4	4	16
Cukup	3	1	3
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	49
		Y	55
		X	11
		Index%	89,09

Butir Keenam

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	5	25
Layak	4	6	24
Cukup	3	0	0
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	49
		Y	55
		X	11
		Index%	89,09

Butir Ketujuh

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	7	35
Layak	4	4	16
Cukup	3	0	0
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	51
		Y	55
		X	11
		Index%	92,73

Butir Kedelapan

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	5	25
Layak	4	6	24
Cukup	3	0	0
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	49
		Y	55
		X	11
		Index%	89,09

Butir Kesembilan

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	4	20
Layak	4	7	28
Cukup	3	0	0
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	48
		Y	55
		X	11
		Index%	87,27

Butir Kesepuluh

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	3	15
Layak	4	8	32
Cukup	3	0	0
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	47
		Y	55
		X	11
		Index%	85,45

Butir Kesebelas

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Layak	5	1	5
Layak	4	8	32
Cukup	3	2	6
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Skor	43
		Y	55
		X	11
		Index%	78,18

$$Total\ Index\% = \frac{Total\ Persentase\ Seluruh\ Butir}{Jumlah\ Butir}$$

Total Index% = 85,70248 %

Kategori Sangat Layak

Lampiran 7. Hasil Uji Coba Luas.

Uji Coba Luas

A. Uji Kemampuan

Butir Pertama (negatif)

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat baik/benar	1	0	0
Baik/benar	2	13	26
Cukup	3	27	81
Kurang	4	6	24
Sangat Kurang	5	3	15
		Total Sko	146
		Y	245
		X	49
		Index%	59,6

Butir Kedua

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat baik	5	3	15
Baik	4	7	28
Cukup	3	21	63
Kurang	2	17	34
Sangat Kurang	1	2	2
		Total Sko	142
		Y	250
		X	50
		Index%	56,8

Butir Ketiga

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat baik	5	3	15
Baik	4	32	128
Cukup	3	9	27
Kurang	2	4	8
Sangat Kurang	1	0	0
		Total Sko	178
		Y	240
		X	48
		Index%	74,2

Butir Keempat

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat baik	5	2	10
Baik	4	16	64
Cukup	3	13	39
Kurang	2	14	28
Sangat Kurang	1	3	3
		Total Sko	144
		Y	240
		X	48
		Index%	60

Butir Kelima

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat baik	5	2	10
Baik	4	29	116
Cukup	3	16	48
Kurang	2	1	2
Sangat Kurang	1	0	0
		Total Sko	176
		Y	240
		X	48
		Index%	73,3

$$Total\ Index\% = \frac{Total\ Persentase\ Seluruh\ Butir}{Jumlah\ Butir}$$

Total Index% = 64,8 %

B. Uji Kemanfaatan

Butir Pertama

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Sesuai/Setuju	5	9	45
Sesuai/Setuju	4	36	144
Cukup	3	4	12
Kurang	2	0	0
Sangat Kurang	1	0	0
		Total Sko	201
		Y	245
		X	49
		Index%	82

Butir kedua

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Sesuai/Setuju	5	3	15
Sesuai/Setuju	4	15	60
Cukup	3	27	81
Kurang	2	3	6
Sangat Kurang	1	1	1
		Total Sko	163
		Y	245
		X	49
		Index%	66,53

Butir Ketiga

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Sesuai/Setuju	5	10	50
Sesuai/Setuju	4	30	120
Cukup	3	8	24
Kurang	2	1	2
Sangat Kurang	1	0	0
		Total Sko	196
		Y	245
		X	49
		Index%	80

Butir Keemat

Jawaban	Skor	Jumlah (orang)	TxPn
Sangat Sesuai/Setuju	5	15	75
Sesuai/Setuju	4	24	96
Cukup	3	9	27
Kurang	2	1	2
Sangat Kurang	1	0	0
		Total Sko	200
		Y	245
		X	49
		Index%	81,63

$$Total\ Index\% = \frac{Total\ Persentase\ Seluruh\ Butir}{Jumlah\ Butir}$$

Total Index % = 77,6 %

Lampiran 8. Hasil Validasi Model.

Validasi Model

A. Model Konseptual

Butir Validasi

Butir Pertama

Jawaban	Skor	Jumlah (orang)	Txpn
ya	2	5	10
Tidak	1	0	0
		Total Skor	10
		Y	10
		X	5
		Index%	100

Butir Kedua

Jawaban	Skor	Jumlah (orang)	Txpn
ya	2	5	10
Tidak	1	0	0
		Total Skor	10
		Y	10
		X	5
		Index%	100

Total Index = 100 %

B. Model Logikal

Butir Verifikasi

Butir Pertama

Jawaban	Skor	Jumlah (orang)	Txpn
ya	2	5	10
Tidak	1	0	0
		Total Skor	10
		Y	10
		X	5
		Index%	100

Butir Kedua

Jawaban	Skor	Jumlah (orang)	Txpn
ya	2	5	10
Tidak	1	0	0
		Total Skor	10
		Y	10
		X	5
		Index%	100

Total Index = 100 %

Butir Validasi

Butir Pertama

Jawaban	Skor	Jumlah (orang)	Txpn
ya	2	5	10
Tidak	1	0	0
		Total Skor	10
		Y	10
		X	5
		Index%	100

Butir Kedua

Jawaban	Skor	Jumlah (orang)	Txpn
ya	2	5	10
Tidak	1	0	0
		Total Skor	10
		Y	10
		X	5
		Index%	100

Total Index = 100 %

C. Model Simulasi

Butir Verifikasi

Butir Pertama

Jawaban	Skor	Jumlah (orang)	Txpn
Ya	2	5	10
Tidak	1	0	0
		Total Skor	10
		Y	10
		X	5
		Index %	100

Butir Kedua (negatif)

Jawaban	Skor	Jumlah (orang)	Txpn
Ya	1	3	3
Tidak	2	2	4
		Total Skor	7
		Y	10
		X	5
		Index %	70
		Total Index =	85 %

Butir Validasi

Butir Pertama

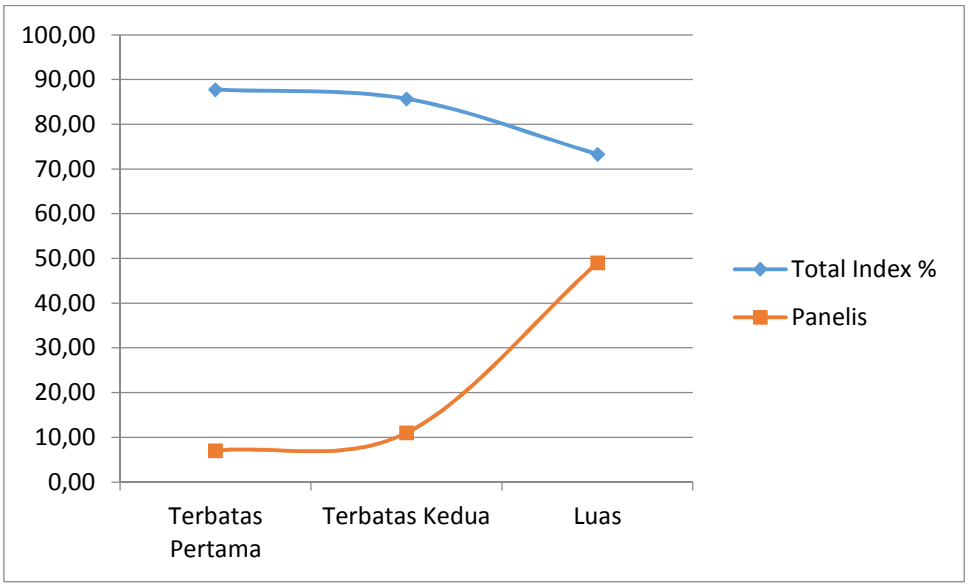
Jawaban	Skor	Jumlah (orang)	Txpn
Ya	2	5	10
Tidak	1	0	0
		Total Skor	10
		Y	10
		X	5
		Index %	100

Butir Kedua

Jawaban	Skor	Jumlah (orang)	Txpn
Ya	2	5	10
Tidak	1	0	0
		Total Skor	10
		Y	10
		X	5
		Index %	100
		Total Index =	100 %

Lampiran 9. Output Hasil Analisis.

Pengujian	Total Index %	Panelis
Terbatas Pertama	87,79	7
Terbatas Kedua	85,70	11
Luas	73,33	49



Lampiran 10. Panduan *Git Server* Manual untuk Instruktur/Dosen.

Git Server Manual

A. Pendahuluan

Git Server merupakan penyedia layanan manajemen kode program yang terdistribusi dalam jaringan. Yaitu dapat digunakan sebagai media belajar bersama mata kuliah pemrograman, yang mana dosen/instruktur dapat memberikan latihan dan tugas pemrograman menggunakan layanan tersebut, sedangkan mahasiswa dapat mengerjakan latihan dan tugas secara kelompok dan terdistribusi dengan layanan ini. Kemudahan layanan ini yaitu (1) keaktifan mahasiswa dalam mengerjakan tugas maupun latihan dapat di log secara *up-to-date* (2) mahasiswa dapat merefisi karya mereka kapanpun (3) menanamkan konsep *social coding* dalam memahami program dan *code* secara terbuka.

B. *Setup* dan Konfigurasi

Persiapan untuk melakukan instalasi dan konfigurasi git, mengikuti langkah – langkah berikut :

1) Koneksi *WiFi*
   Koneksikan *workstation* anda ke *Access Point* dengan *SSID* "**gitcahunp**", kemudian *workstation* anda akan mendapatkan *ip address* secara automatis.
2) *Download git*
   Untuk men-*download git* yang telah disediakan, caranya buka *browser* kemudian akses alamat *url* `http://git.gitserver/download/` *download git versi* 1.8. atau anda bisa men-*download* versi terbaru di : `https://msysgit.github.com/`
3) Instalasi
   Untuk melakukan instalasi *git* ikuti sesuai instruksi, atau anda bisa men-*download* panduan instalasi *git* yang telah di sediakan pada alamat *url* : `http://git.gitserver/download/`.
4) Konfigurasi
   Setelah melakukan instalasi maka tahapan yang pertama kali dilakukan adalah melakukan konfigurasi awal, ini hanya dilakukan sekali setelah melakukan instalasi, yaitu dengan cara :
   - Buka Aplikasi *Git Bash*
   - Ketikkan perintah berikut :

     ```
     $ git config --global user.name "Nama Lengkap"
     $ git config --global user.email "user@email.com"
     ```

C. Mendaftarkan `public_key`

1) Generating `ssh-keygen`

Caranya, Buka aplikasi *Git Bash*

```
$ ssh-keygen -t rsa -C "alamat_email/nama_hostname"
```

Kosongi bagian *passphrase* dengan cara di enter.

2) Daftarkan `public_key`

```
$ cd ~/.ssh/
$ cp id_rsa.pub <nicname>.pub
$ scp <nickname>.pub dosen@git.gitserver:/tmp/
```

Password @dosenunp

Kontak administrator *Gitolite*, hingga di konfigurasi `public_key` anda.

D. Konfigurasi *gitolite-admin*

Clone *Gitolite-admin* dengan perintah berikut :

```
$ cd
$ git clone git@git.gitserver:gitolite-admin.git
$ cd gitolite-admin
$ notepad.exe conf/gitolite.conf
```

Tambahkan *script* berikut ini :

```
    repo /dosen/[a-z].*
          RW+   =  <nickname> // sesuai nama <nickname>.pub
          C     =  <nickname>
```

Simpan, kemudian anda lakukan *commit* :

```
$ git add conf/gitolite.conf
$ git commit -m "Pesan Perubahan"
$ git push origin master
```

Membuat *repository* baru seperti berikut :

```
$ cd
$ git clone git@git.gitserver:/dosen/<nama_project>.git
// contoh :
$ git clone git@git.gitserver:/dosen/latihan-java-oop.git
$ cd latihan-java-oop.git
$ git status
```

Pada tahapan ini *file* program *java* bisa di kopikan atau di buat dalam *directory* ini, kemudian anda bisa melakukan *commit* :

```
$ git add --all
$ git commit -m "Pesan Perubahan"
$ git push origin master
```

Konfigurasi *gitolite-admin* untuk publikasi latihan dan tugas.

```
$ cd
$ cd gitolite-admin.git
$ notepad.exe conf/gitolite.conf
```

Tambahkan *script* berikut :

```
    repo dosen/latihan-java-oop
          RW+   =  @all
```

```
desc  =  "Repository Latihan Java OOP”
R     =  gitweb daemon
```

Simpan, dan kemudian anda lakukan *commit* :

```
$ git add conf/gitolite.conf
$ git commit –m "Pesan Perubahan"
$ git push origin master
```

E. Akses *Repository*

Mahasiswa dapat mengakses *repository* `latihan-java-oop.git` dengan mengakses alamat *url* `http://www.gitserver/`
Kemudian mahasiswa dapat melihat *repository*-nya maupun menggandakan *repository* `latihan-java-oop.git` dengan perintah :

```
$ git clone http://git.gitserver/dosen/latihan-java-oop.git
```

Hanya saja perintah *clone* yang menggunkan *protocol http* tidak bisa digunakan untuk akses *read-write*. Untuk mendapatkan akses *read-write* maka harus menggunkan protocol *ssh* dengan mendaftarkan `public_key` kepada dosen/instruktur-nya masing – masing.

F. Menkonfigurasi `public_key` mahasiswa

Workstation mahasiswa :
Generating *ssh-keygen*

```
$ ssh-keygen –t rsa –C "alamat_email/nama_hostname"
```

Kosongi bagian *passphrase* dengan cara di enter.

```
$ cd ~/.ssh/
$ cp id_rsa.pub <nickname>.pub
$ scp <nickname>.pub student@git.gitserver:/tmp/
```

Password @student

Workstation Dosen/instruktur:

```
$ ssh dosen@git.gitserver
```

Password @dosenunp

```
$ sudo mv /tmp/*.pub /home/git/repositories/key-mahasiswa/
```

Akses alamat *url* `http://git.gitserver/key-mahasiswa/`
Download semua `public_key` kemudian disimpan ke *path* :

```
C:/User/<username>/gitolit-admin/keydir/kelas-2a/
```

Kemudian anda bisa melakukan *commit* pada *gitolite-admin*:

```
$ exit
$ cd
$ cd gitolite-admin.git
$ git add --all
$ git commit –m "Pesan Perubahan"
```

Konfigurasi *gitolite-admin* untuk memberi hak akses mahasiswa.

```
$ notepad.exe conf/gitolite.cont
```

Tambahkan *script* berikut untuk mendefinisakan *variable* `kelas-2a`:

```
    @kelas-2a = alice tomi bob etc...
```

Tambahkan *scipt* berikut untuk mengijinkan mahasiswa melakukan *create repository* pada *directory* mahasiswa :

```
    repo mahasiswa/[a-z].*
          RW+    = @all
          C      = @kelas-2a
```

Maka dengan demikian mahasiswa bisa *create repository* untuk mengumpulkan tugas maupun belajar bersama secara kolaboratife sesuai kelompoknya masing – masing.

Lampiran 11. Panduan *Git Server* Manual untuk Mahasiswa.

Git Server Manual

A. Pendahuluan

Git Server merupakan penyedia layanan manajemen kode program yang terdistribusi dalam jaringan. yang mana dapat digunakan sebagai media belajar bersama mata kuliah pemrograman, yang mana mahasiswa dapat mengerjakan latihan dan tugas secara kelompok dan terdistribusi dengan layanan ini. Kemudahan layanan ini yaitu (1) mahasiswa dapat mengerjakan tugas maupun latihan dapat di log secara *up-to-date* (2) mahasiswa dapat merefisi karya mereka kapanpun (3) menanamkan konsep *social coding* dalam memahami program dan *code* secara terbuka.

B. *Setup* dan Konfigurasi

Persiapan untuk melakukan instalasi dan konfigurasi git, mengikuti langkah – langkah berikut :

1) Koneksi *WiFi*
   Koneksikan *workstation* anda ke *Access Point* dengan *SSID* "**gitcahunp**", kemudian *workstation* anda akan mendapatkan *ip address* secara automatis.
2) *Download git*
   Untuk men-*download git* yang telah disediakan, caranya buka *browser* kemudian akses alamat *url* `http://git.gitserver/download/` *download git versi* 1.8. atau anda bisa men-*download* versi terbaru di :
   `https://msysgit.github.com/`
3) Instalasi
   Untuk melakukan instalasi *git* ikuti sesuai instruksi, atau anda bisa men-*download* panduan instalasi *git* yang telah di sediakan pada alamat *url* :
   `http://git.gitserver/download/`.
4) Konfigurasi
   Setelah melakukan instalasi maka tahapan yang pertama kali dilakukan adalah melakukan konfigurasi awal, ini hanya dilakukan sekali setelah melakukan instalasi, yaitu dengan cara :
   - Buka Aplikasi *Git Bash*
   - Ketikkan perintah berikut :

     ```
     $ git config --global user.name "Nama Lengkap"
     $ git config --global user.email "user@email.com"
     ```

C. Mendaftarkan `public_key`

1) Generating `ssh-keygen`
   Caranya, Buka aplikasi *Git Bash*

   ```
   $ cd
   $ ssh-keygen –t rsa –C "alamat_email/nama_hostname"
   ```

   Kosongi bagian *passphrase* dengan cara di enter sebanyak dua kali.
2) Daftarkan `public_key`

```
$ cd ~/.ssh/
$ cp id_rsa.pub <nicname>.pub
$ scp <nickname>.pub student@git.gitserver:/tmp/
```

Password @student

Kontak administrator *Gitolite*, hingga di konfigurasi `public_key` anda.

D. Akses *Repository*

Sekarang anda dapat mengakses *repository* dengan mengakses alamat *url* `http://www.gitserver/`

Kemudian anda dapat menggandakan *repository* dengan perintah :

```
$ cd
$ git clone http://git.gitserver/latihan-git.git
```

Hanya saja perintah *clone* yang menggunkan *protocol http* tidak bisa digunakan untuk akses *read-write*. Untuk mendapatkan akses *read-write* maka harus menggunkan protocol *ssh* dengan mendaftarkan `public_key` kepada dosen/instruktur-nya masing – masing.

Untuk menggandakan repository melalui protokol SSH lakukan perintah berikut :

```
$ cd
$ rm –rf latihan-git/
$ git clone git@git.gitserver:/latihan-git.git
```

E. Mengoperasikan *repository* `latihan-git.git`

Cara mengoperasikan *repository* `latihan-git.git`, ikuti perintah berikut :

```
$ cd latihan-git.git
$ notepad.exe README.md
```

Tambahkan nama anda ada *header* huruf yang telah di sediakan, contoh :

```
M
---
* M. Saiful Mukharom
[GitHub](https://github.com/saifulindo),[@saifulindo](https:
//twitter.com/saifulindo))
```

Simpan, Berikutnya lakukan `commit` :

```
$ git add README.md
$ git commit –m "Pesan Perubahan"
$ git pull origin master
$ git push origin master
```

Untuk memeriksa hasil perubahan yang telah anda lakukan, akses alamat *url* `http://www.gitserver/latihan-git.git` jika terjadi konflik pada baris yang sama maka salah satu membetulkan dengan cara komunikasi kepada nama mahasiswa lain yang terlibat konflik. Selanjutnya setelah dimodifikasi lakukan `git add`, `commit`, `pull`, dan `push` seperti perintah sebelumnya.

F. Kesimpulan

Git Server Manual ini adalah panduan untuk anda ketika sedang latihan mengoperasikan git, ketika anda mengerjakan tugas ataupun latihan pada mata kuliah pemrograman pastikan anda sudah sukses ketika mengoperasikan *repository* `latihan-git.git`, demikian panduan ini dibuat untuk memudahkan anda mengoperasikan *git*.

Lampiran 12. Surat Keterangan Pelakasanaan Penelitian.

UNIVERSITAS NUSANTARA PGRI KEDIRI
LEMBAGA PENELETIAN (LEMLIT)
Alamat : Kampus I Jl. KH. Achmad Dahlan No. 76 Kediri (64112)
Telp. : (0354) 771576, 771503, 771495 Fax. (0354) 771576
website : www.unpkediri.ac.id, email : lemlit.unpkediri@gmail.com

Nomor : 037 /lemlit-UNP/2014
Lampiran : 1 (satu) berkas
Hal : Permohonan ijin melakukan penelitian

Yth.
Dekan Fakultas Teknik
Universitas Nusantara PGRI Kediri
Di
Tempat

Dengan ini kami hadapkan mahasiswa Universitas Nusantara PGRI Kediri :

Nama	: M. Saiful Mukharom
NPM	: 09.1.03.02.0299
Fak./Jur./Prodi.	: Teknik/S1 Teknik Informatika
Maksud	: Ijin melakukan penelitian untuk penulisan Skripsi/Tugas Akhir
Judul	: Implementasi *Git Server* dengan Pendekatan *GitHub Social Coding* dalam Peningkatan Pembelajaran Mahasiswa

Sehubungan dengan hal tersebut, kami mohon bantuannya untuk memberi ijin kepada mahasiswa yang bersangkutan guna mendapatkan data – data penelitian pada lembaga yang bapak/ibu/sdr. Pimpin sebagai bahan penulisan Skripsi/Tugas Akhir.

Atas perhatian dan bantuannya, kami ucapkan terima kasih.

Kediri, 23 April 2014
a.n. Rektor
Ketua LEMLIT

Dr. SURYANTO, M.Si.

Tembusan :
1. Rektor
2. Dekan Fakultas

UNIVERSITAS NUSANTARA PGRI KEDIRI
FAKULTAS TEKNIK
Program Studi Teknik Informatika
Alamat : Kampus I Jl. KH. Achmad Dahlan No. 76 Kediri (64112)
Telp. : (0354) 771576, 771503, 771495 Fax. (0354) 771576

SURAT KETERANGAN

Nomor: 0013 / FT-UNP/TI/C/V/2015

Yang bertanda tangan dibawah ini, Ketua Program Studi Teknik Informatika Universitas Nusantara PGRI Kediri :

Nama	: Ahmad Bagus Setiawan, S.T, MM.
Jabatan	: Ketua Program Studi Teknik Informatika

Menerangkan bahwa :

Nama	: M. Saiful Mukharom
NPM	: 09.1.03.02.0299
Fak./Jur./Prodi.	: Teknik/S1 Teknik Informatika
Perguruan Tinggi	: Universitas Nusantara PGRI Kediri

Yang bersangkutan benar – benar telah melaksanakan penelitian pada tanggal 13 April 2015 – 18 Mei 2015, dalam rangka penyusunan skripsi yang berjudul : **"Merancang *Git Server* dengan Pendekatan *GitHub Social Coding* dalam Peningkatan Pembelajaran Mahasiswa"**. Pada mata kuliah Rekayasa Perangkat Lunak dan Pemrograman Web yang diampu oleh :

Nama	: Riski Aswi Ramadhani, S.Kom
Jabatan	: Dosen

Demikian surat ini dibuat dan digunakan sebagaimana mestinya.

Kediri, 31 Mei 2015
Ketua Program Studi Teknik Informatika,

Ahmad Bagus Setiawan, S.T, MM.

Mengetahui,
Dosen Pengampu Mata Kuliah.

Riski Aswi Ramadhani, S.Kom

Lampiran 13. Berita Acara Kemajuan Pembimbingan.

PERSETUJUAN BAU

BERITA ACARA
KEMAJUAN PEMBIMBINGAN
PENULISAN KARYA TULIS ILMIAH

1. NAMA MAHASISWA : M. SAIFUL MUKHAROM
   NPM : 09.1.03.02.0299
   Fak/Jur/Prodi : Teknik / S1 Teknik Informatika
   Alamat Rumah : Jl. Ngadirimo Gg II, RT/RW
   Alamat email : fithesaif@gmail.com
   No. Telp. / HP : 0354 - 687086 / 085730947129

2. DOSEN PEMBIMBING I : Dr. Muchson, SE., M.M
   Alamat Rumah : Tamanan, Kota Kediri
   Alamat email : -
   No. Telp. / HP. : 081359155189

3. DOSEN PEMBIMBING II : Ari Eka Prasetyanto, S.Kom
   Alamat Rumah : -
   Alamat email : -
   No. Telp. / HP : 085631777772

4. JUDUL KTI :
   Merancang Git Server dengan Pembelajaran GitHub Social Coding Dalam Peningkatan Pembelajaran Mahasiswa

Catatan :
1. Periode Bimbingan (Sesuai SK Rektor) :
2. Jadwal Bimbingan :

	Hari	Pukul	Tempat / Ruang
Pembimbing I			
Pembimbing II			

3. Kemajuan Bimbingan :

Pembimbing I

NO.	TANGGAL	MATERI	MASALAH	TT. DOSEN
	16-8-2014	Bab I — Bab V	Latar belakang, rumusan,	
			tujuan, Manfaat, ACC	
		Bab II	ACC Maju	
			ACC	
		Bab III	Metode Penelitian	
			ACC	
			Pembahasan Revisi	
			ACC	
		9-9-2014	Lulus ujian - Revisi	

Pembimbing II

NO.	TANGGAL	MATERI	MASALAH	TT. DOSEN
1		Metodologi		
2		Landasan -		
3		Teori		
4		Analisa Perancangan		
5		Implementasi & Evaluasi		
6	14/8/14	testing	ACC sidang	

Mengetahui,
Kaprodi

Ahmad Bagus Setiawan
NIDN 0703018704

Kediri, 20 Mei 2015
Mahasiswa Ybs,

M. Saiful Mukhrom
NIDN.

Lampiran 14. Dokumentasi Kegiatan Penelitian.

Pengujian Terbatas Pertama

Pengujian Terbatas Kedua

Pengujian Terbatas Kedua

Pengujian Luas

Pengujian Luas

Lampiran 15. Acc Revisi Ujian/Sidang Skripsi.

UNIVERSITAS NUSANTARA PGRI KEDIRI

FAKULTAS TEKNIK

Program Studi :Teknik Mesin, Teknik Elektro, Teknik Industri,

Teknik Informatika, Sistem Informasi

Alamat : JL. K.H. Achmad Dahlan No. 76 Telp (0354) 776706 Kediri

LEMBAR REVISI UJIAN SKRIPSI

Nama : M. Saiful Mukharom

NPM : 09.1.03.02.0299

Judul Skripsi : Merancang Git Server dengan Pendekatan GitHub Social Coding dalam Peningkatan Pembelajaran Mahasiswa

No.	Komponen
1.	Relevansi Judul dengan Perkembangan Teknologi Informasi
	Saran Perbaikan :
2.	Kesesuaian Teori yang digunakan
	Saran Perbaikan :
3.	Metodologi
	Saran Perbaikan :
4.	Sistematika Penulisan dan Bahasa Ilmiah
	Saran Perbaikan : Kutipan Gambar Halaman, tolong diperbaiki, Rumusan masalah → Jumlah dan Kesimpulan harus mengalir!
5.	Penguasaan Bahasa Pemrograman yang digunakan
	Saran Perbaikan :
6.	Keamanan Program
	Saran Perbaikan :
7.	Penguasaan dalam Pengujian Program
	Saran Perbaikan :
8.	Lain - Lain
	Saran Perbaikan :

ACC Revisi

Kediri, 28-5-2015

Penguji I

Kediri, 28 Mei 2015

Penguji I

Penguji I

UNIVERSITAS NUSANTARA PGRI KEDIRI

FAKULTAS TEKNIK

Program Studi :Teknik Mesin, Teknik Elektro, Teknik Industri,
Teknik Informatika, Sistem Informasi
Alamat : JL. K.H. Achmad Dahlan No. 76 Telp (0354) 776706 Kediri

LEMBAR REVISI UJIAN SKRIPSI

Nama : M. Saiful Mukharom

NPM : 09.1.03.02.0299

Judul Skripsi : Merancang Git Server dengan Pendekatan GitHub Social Coding dalam Peningkatan Pembelajaran Mahasiswa

No.	Komponen
1.	Relevansi Judul dengan Perkembangan Teknologi Informasi
	Saran Perbaikan :
2.	Kesesuaian Teori yang digunakan
	Saran Perbaikan : - cukup
3.	Metodologi
	Saran Perbaikan :
4.	Sistematika Penulisan dan Bahasa Ilmiah
	Saran Perbaikan : - abstrak, daftar pustaka. - Semua gambar, tabel/ dan diagram harus punya paragraf penjelas/cerita - Setelah gambar 3..... cukup 1 paragraf. (Tabel - Contoh).
5.	Penguasaan Bahasa Pemrograman yang digunakan
	Saran Perbaikan :
6.	Keamanan Program
	Saran Perbaikan :
7.	Penguasaan dalam Pengujian Program
	Saran Perbaikan :
8.	Lain - Lain spasi naskah - tata tulis
	Saran Perbaikan : di Buku Laporan.

ACC Revisi

Kediri, 5 Juni 2015

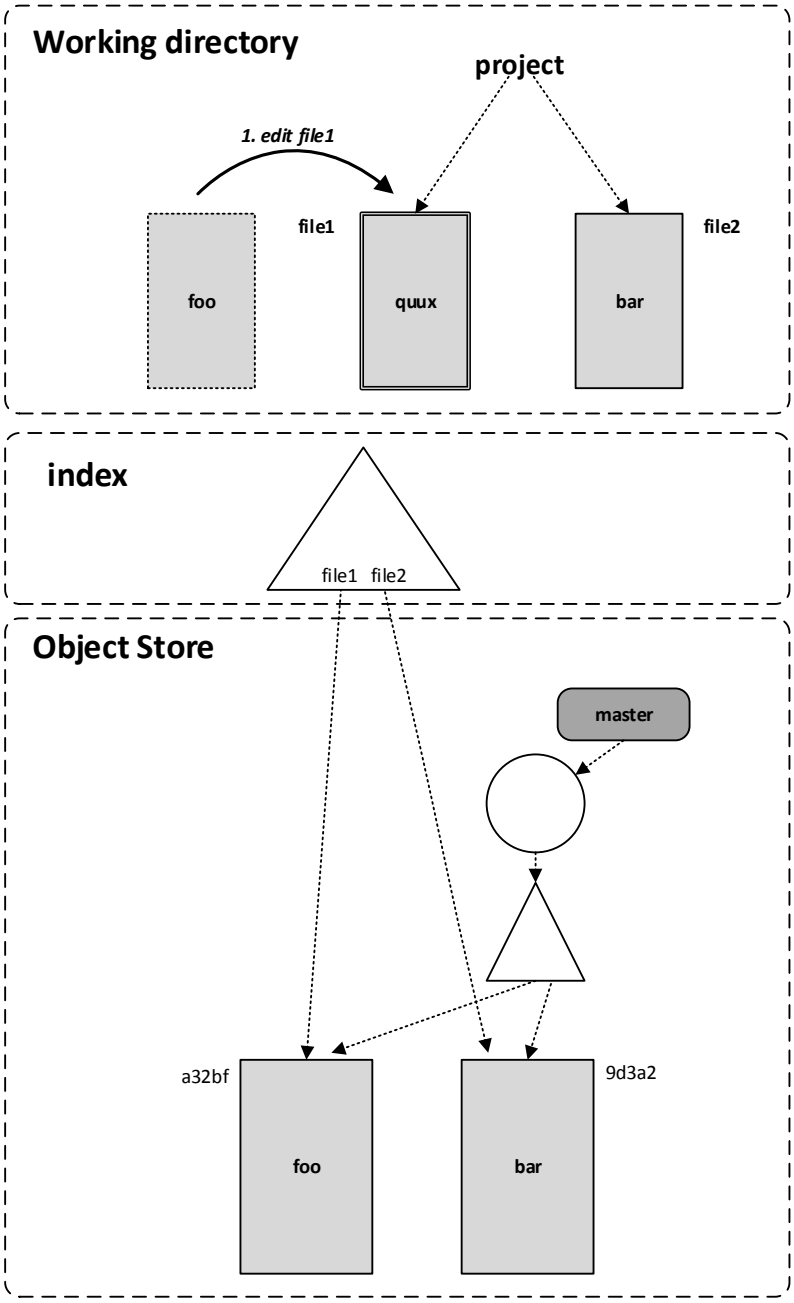
Penguji II

Kediri, 28 Mei 2015

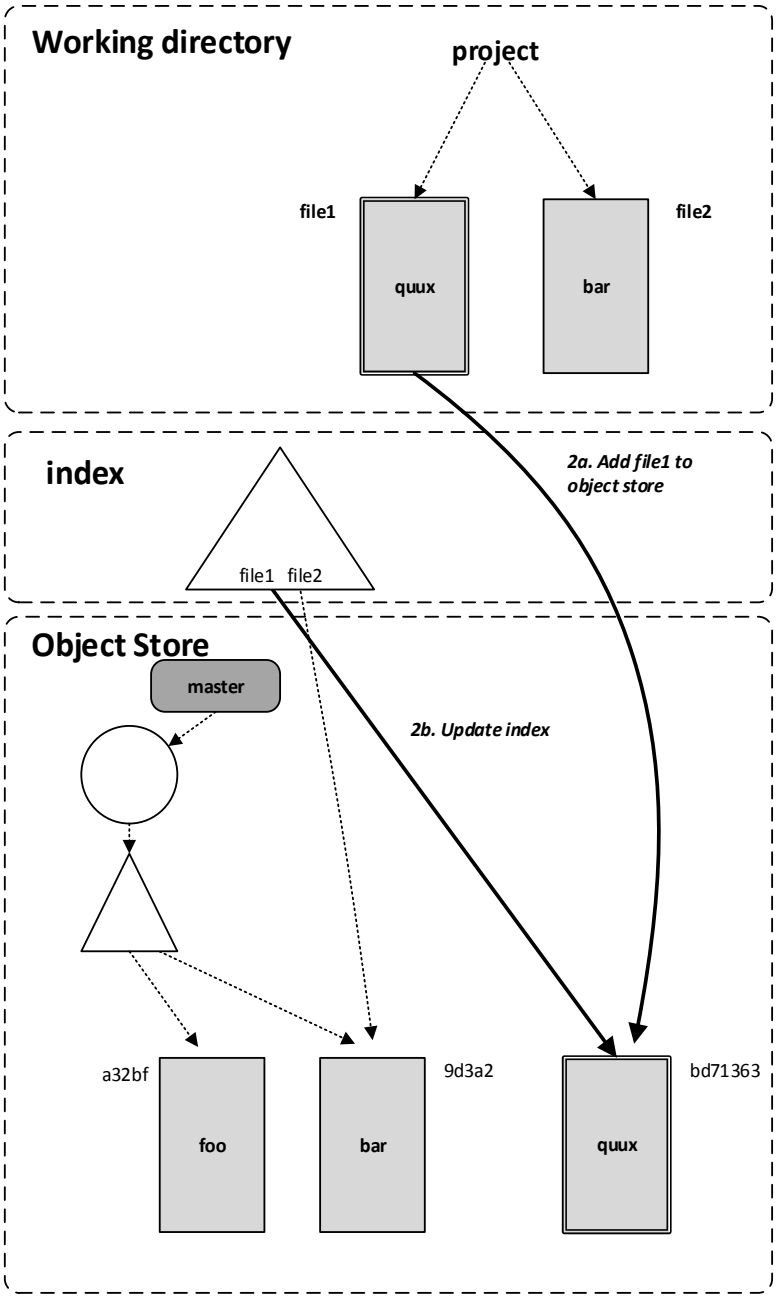
Penguji II

Penguji II

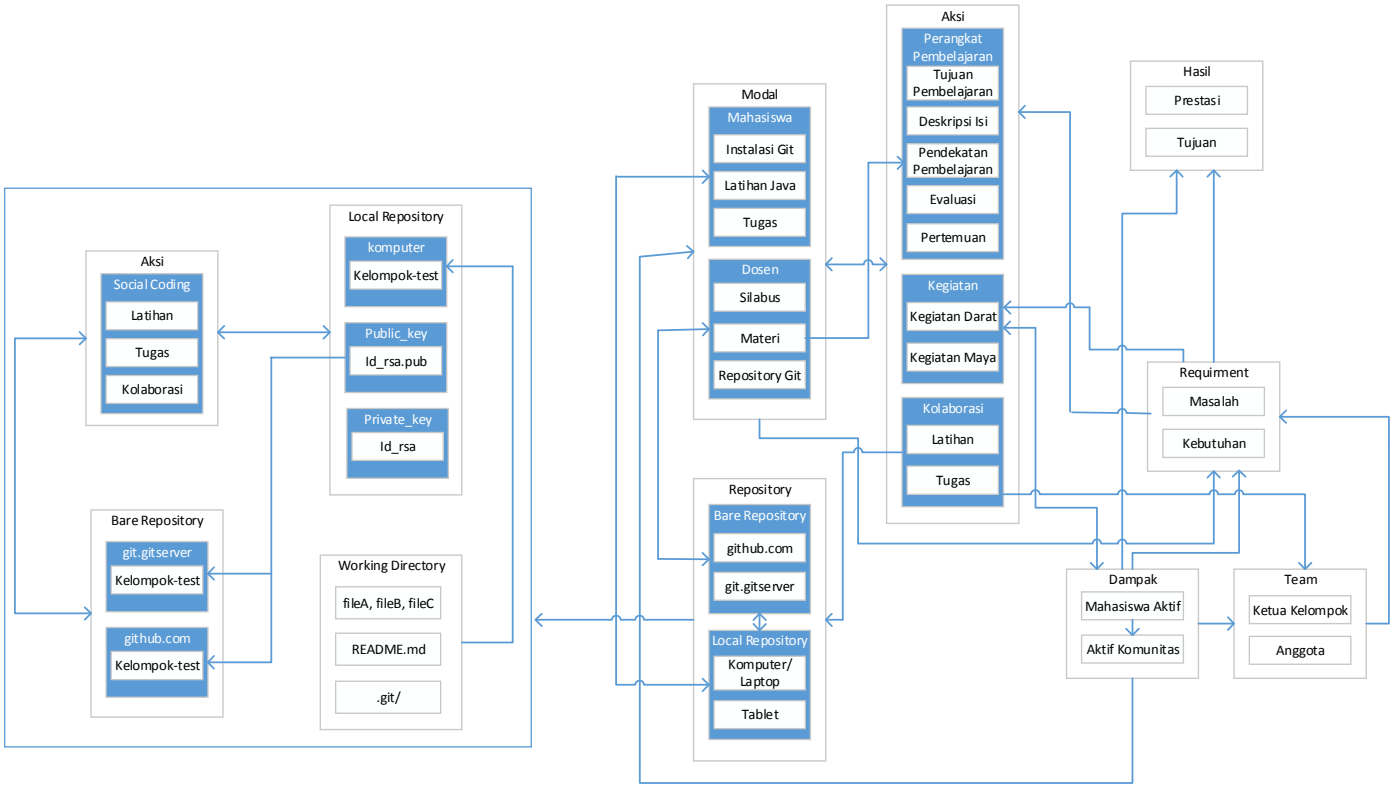
Lampiran 16. Gambar-gambar



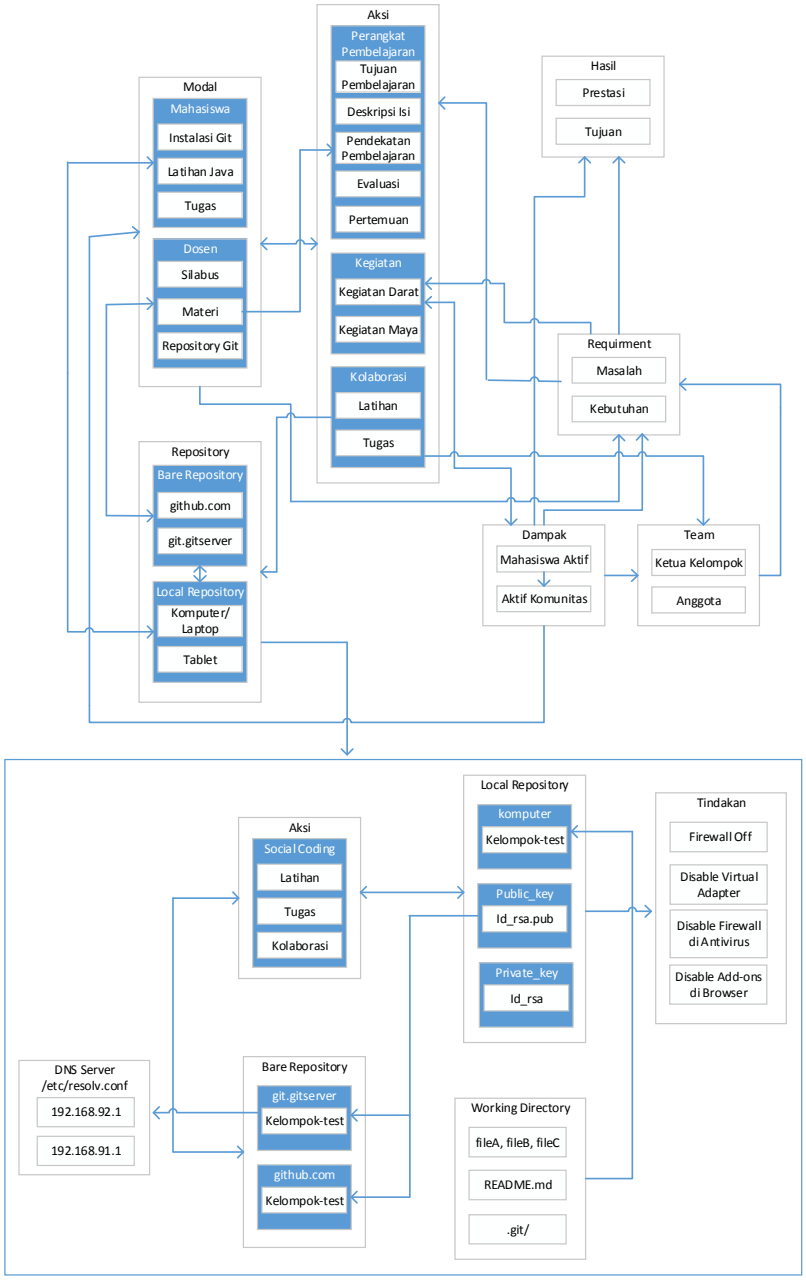
Gambar 3.19. Setelah *Editing File1* (Loeliger, 2012:61)



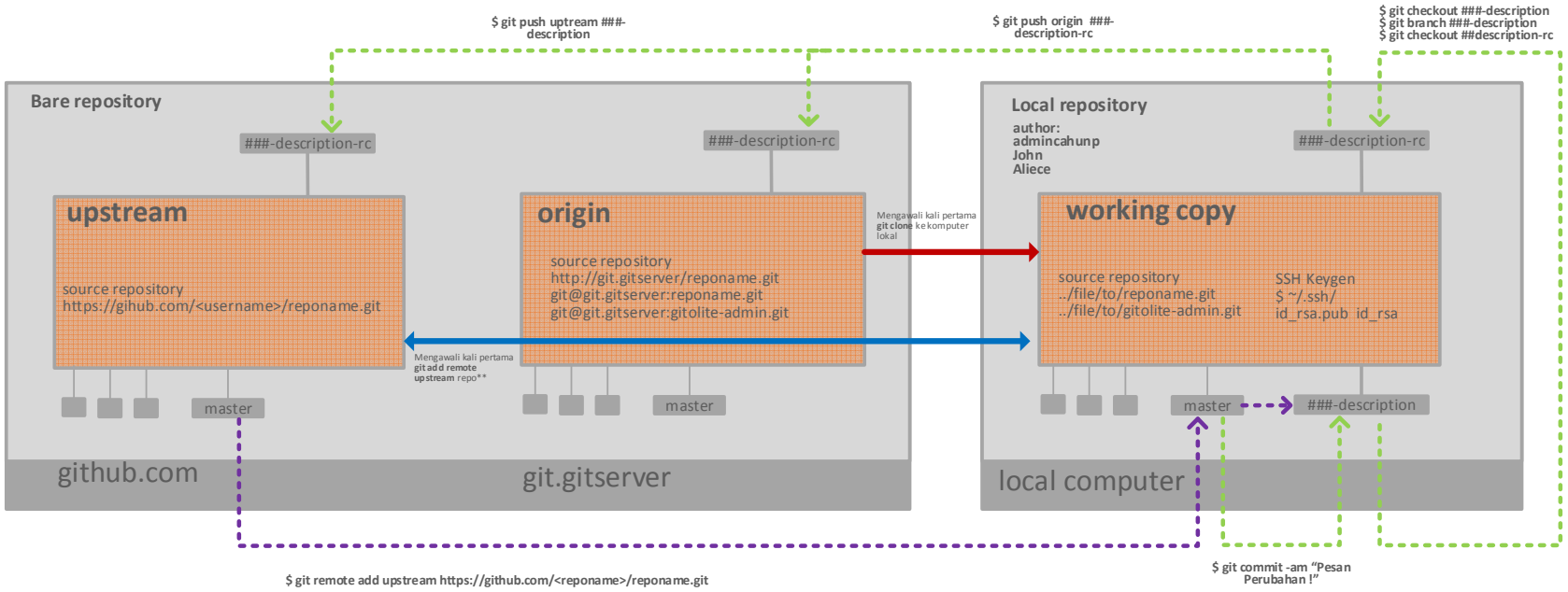
Gambar 3.20 Setelah Git Add (Loeliger, 2012:63)



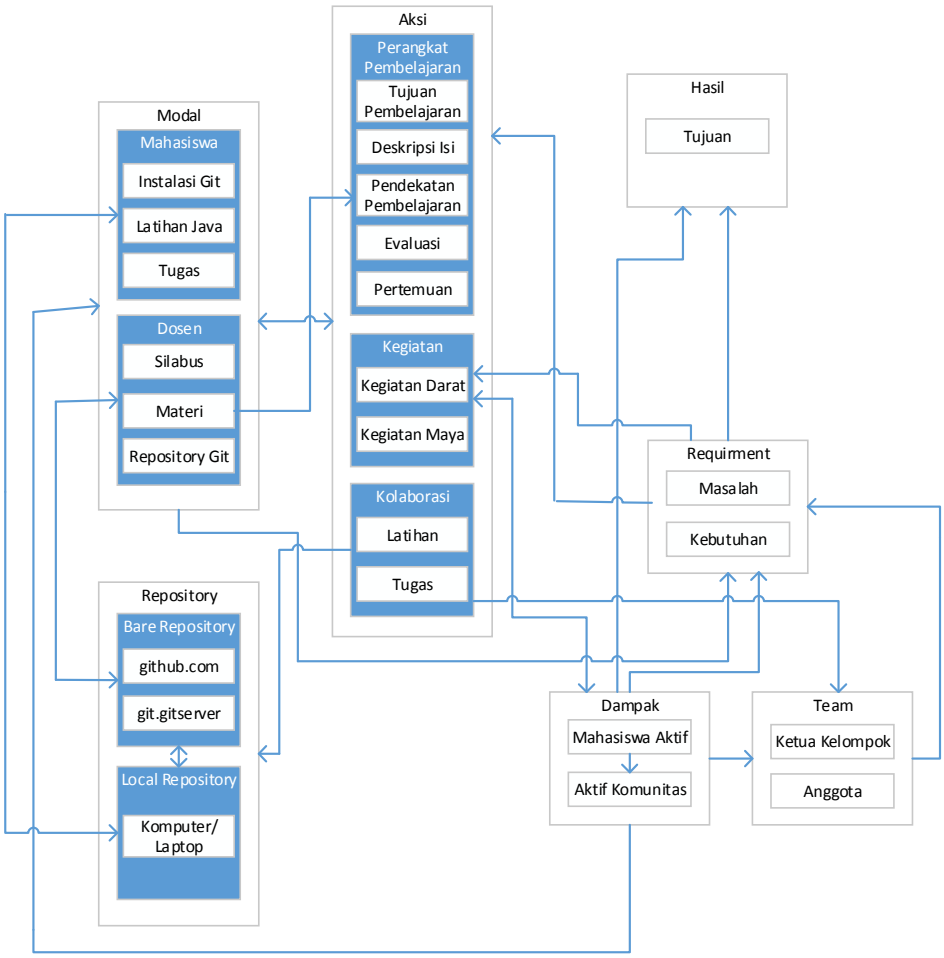
Gambar 4.1. Diagram Model Pembelajaran Social Coding dengan Diagram Uji Kelayakan.



Gambar 4.3. Diagram Model Pembelajaran Social Coding Hipotetik.



Gambar 4.4. Model Simulasi Komputer.



Gambar 4.5. Diagram Model Pembelajaran Social Coding.

UNIVERSITAS NUSANTARA PGRI KEDIRI
FAKULTAS TEKNIK
Program Studi Teknik Informatika
Alamat : Kampus I Jl. KH. Achmad Dahlan No. 76 Kediri (64112)
Telp. : (0354) 771576, 771503, 771495 Fax. (0354) 771576

SURAT KETERANGAN

Nomot :

Yang bertanda tangan dibawah ini, Ketua Program Studi Teknik Informatika Universitas Nusantara PGRI Kediri :

Nama	: Ahmad Bagus Setiawan, S.T, MM.
Jabatan	: Ketua Program Studi Teknik Informatika

Menerangkan bahwa :

Nama	: M. Saiful Mukharom
NPM	: 09.1.03.02.0299
Fak./Jur./Prodi.	: Teknik/S1 Teknik Informatika
Perguruan Tinggi	: Universitas Nusantara PGRI Kediri

Yang bersangkutan benar – benar telah melaksanakan penelitian pada tanggal 13 April 2015 – 18 Mei 2015, dalam rangka penyusunan skripsi yang berjudul : **"Merancang *Git Server* dengan Pendekatan *GitHub Social Coding* dalam Peningkatan Pembelajaran Mahasiswa"**. Pada mata kuliah Rekayasa Perangkat Lunak dan Pemrograman Web yang diampu oleh :

Nama	: Riski Aswi Ramadhani, S.Kom
Jabatan	: Dosen

Demikian surat ini dibuat dan digunakan sebagaimana mestinya.

Kediri, 31 Mei 2015	Mengetahui,
Ketua Program Studi Teknik Informatika.	Dosen Pengampu Mata Kuliah.
Ahmad Bagus Setiawan, S.T, MM.	Riski Aswi Ramadhani, S.Kom

BERITA ACARA
KEMAJUAN PEMBIMBINGAN
PENULISAN KARYA TULIS ILMIAH

1. NAMA MAHASISWA : M. Saiful Mukharom
   NPM : 09.1.03.02.0299
   Fak/Jur/Prodi. : Teknik/SI Teknik Informatika
   Alamat Rumah : Jl. Ngadisimo Gg II, RT/RW 04/07, Kel. Ngadiredjo, Kec. Kota, Kediri
   Alamat email : ritnesaif@gmail.com
   No. Telp. / HP : 0354-697086/085730947129

2. DOSEN PEMBIMBING I : Dr. M. Muchson, SE., M.M
   Alamat rumah : -
   Alamat email : -
   No. Telp. / HP : 081359155189

3. DOSEN PEMBIMBING II : Ari Eka Prasetiyanto, S.Kom
   Alamat rumah : -
   Alamat email : -
   No. Telp./ HP : 08563177772

4. JUDUL KTI : Implementasi *Git Server* dengan Pendektan *GitHub Social Coding* dalam Peningkatan Pembelajaran Mahasiswa

Catatan :

1. Periode Bimbingan (sesuai SK Rektor) :
2. Judul Bimbingan

	Hari	Pukul	Tempat/Ruang
Pembimbing I			
Pembimbing II			

3. Kemajuan Bimbingan :

Pembimbing I

No.	Tanggal	Materi	Catatan	Paraf

Mengetahui,
Kaprodi

Ahmad Bagus Setiawan, S.T, MM.
NIDN.

Kediri,
Mahasiswa Ybs.

M. Saiful Mukharom
NPM. 09.1.03.02.0299

Pembimbing II

No.	Tanggal	Materi	Catatan	Paraf

Mengetahui,
Kaprodi

Ahmad Bagus Setiawan, S.T, MM.
NIDN.

Kediri,
Mahasiswa Ybs.

M. Saiful Mukharom
NPM. 09.1.03.02.0299

UNIVERSITAS NUSANTARA PGRI KEDIRI
LEMBAGA PENELETIAN (LEMLIT)
Alamat : Kampus I Jl. KH. Achmad Dahlan No. 76 Kediri (64112)
Telp. : (0354) 771576, 771503, 771495 Fax. (0354) 771576
website : www.unpkediri.ac.id, email : lemlit.unpkediri@gmail.com

Nomor : /lemlit-UNP/2014
Lampiran : -
Hal : Permohonan ijin melakukan penelitian

Yth. **Rini Indriati, M. Kom.**
Dekan Fakultas Teknik
Universitas Nusantara PGRI Kediri
Di
Tempat

Dengan ini kami hadapkan mahasiswa Universitas Nusantara PGRI Kediri :

Nama	: M. Saiful Mukharom
NPM	: 09.1.03.02.0299
Fak./Jur./Prodi.	: Teknik/S1 Teknik Informatika
Maksud	: Ijin melakukan penelitian untuk penulisan Skripsi/Tugas Akhir
Judul	: Implementasi *Git Server* dengan Pendekatan *GitHub Social Coding* dalam Peningkatan Pembelajaran Mahasiswa

Sehubungan dengan hal tersebut, kami mohon bantuannya untuk memberi ijin kepada mahasiswa yang bersangkutan guna mendapatkan data – data penelitian pada lembaga yang bapak/ibu/sdr. Pimpin sebagai bahan penulisan Skripsi/Tugas Akhir.

Atas perhatian dan bantuannya, kami ucapkan terima kasih.

Kediri, 23 April 2014
a.n. Rektor
Ketua LEMLIT

Dr. SURYANTO, M.Si.

Tembusan :
1. Rektor
2. Dekan Fakultas

PENILAIAN UJIAN/SIDANG
KARYA TULIS ILMIAH

NAMA MAHASISWA : M. SAIFUL MUKHAROM
NPM : 09.1.03.02.0299
Fak./Jur./Prodi. : Teknik/
Judul KTI : Implementasi *Git Server* dengan Pendekatan *GitHubSocial Coding* dalam peningkatan Pembelajaran Mahasiswa

No	Aspek yang Dinilai	Skor
1	Sistematika dan Teknik Penulisan (boot maksimal 5). a. Ketepatan Penulisan Judul	
	b. Kelengkapan Bagian awal (hal judul, persetujuan, pengesahan, pernyataan, kata pengantar, abstrak, daftar isi, daftar tabel, daftar gambar, daftar lampiran)	
	c. Kelengkapan bagian akhir (daftar pustaka, lampiran : instrumen, proses dan hasil analisis data, surat-surat penelitian)	
	d. Teknik penulisan/pengetikan sesuai pedoman	
	e. Konsistensi (penggunaan bahasa, istilah, simbol, dll).	
2.	Kesesuaian isi masing – masing Bab : a. Bab I (bobot maksimal 10)	
	b. Bab II (bobot maksimal 10)	
	c. Bab III (bobot maksimal 10)	
	d. Bab IV (bobot maksimal 10)	
	e. Bab V (bobot maksimal 10)	
3.	Presentasi dan argumen berpikir ilmiah dalam mengemukakan pendapat (bobot maksimal 20) a. Presentasi (kejelasan penyajian konsep dan teori, sistematika penyajian, penggunaan bahasa lisan, penekanan hal penting, penggunaan waktu, penggunaan teknologi)	
	b. Argumentasi (kemampuan mengemukakan pendapat, ketepatan jawaban, jujur/terbuka menerima saran, pengendalian emosi)	
4.	Relevansi permasalahan penelitian dengan bidang ilmu/prodi./jurusan (bobot maksimal 5)	
5.	Implikasi dan kegunaan hasil penelitian yang dipetik (bobot maksimal 10)	
6.	Orisinalitas dan kebaharuan permasalahan penelitian (bobot maksimal 10)	
	Jumlah	

Kediri,
Penguji

NIDN.