


# Dasar-Dasar Menggunakan GIT

By Arul Ferian  
Ramadloni

# Apa Tujuannya???

- Mengenal perintah yang akan digunakan ketika menggunakan Git
  - Dapat mengkonfigurasi dan inisialisasi repository
  - Bagaimana cara tracking files, cara stage, dan commit file
  - Mengatur git tentang otoritas tertentu
  - Cara membatalkan kesalahan dengan mudah
  - Cara menelusuri riwayat perubahan commit, push maupun pull
- 

# Git Repository

- **Cara Mendapatkannya** adalah 1) dari local directory yang tidak dibawah version control kemudian mengubah menjadi Git Repo atau 2) cloning dari repository lain
- **Cara Memulainya** *Jika belum terdapat pada version control dan ingin melakukan controlling git maka **pergi ke directory terlebih dahulu***

```
$ cd /home/user/my_project
```

```
C:\Users\ASUS>git init  
Initialized empty Git repository in C:/Users/ASUS/.git/
```

- **Jika sudah berada pada file yang sudah berada version control, pertama, tracking file dan kemudian mencoba commit**

```
$ git add *.c  
$ git add LICENSE  
$ git commit -m 'Initial project version'
```

## Cloning Repository

```
C:\Users\ASUS>git clone https://github.com/libgit2/libgit2
Cloning into 'libgit2'...
remote: Enumerating objects: 99269, done.
remote: Counting objects: 100% (99269/99269), done.
remote: Compressing objects: 100% (26742/26742), done.
Receiving objects: 27% (26803/99269), 12.15 MiB | 514.00 KiB/s
```

```
$ git clone https://github.com/libgit2/libgit2 mylibgit
```

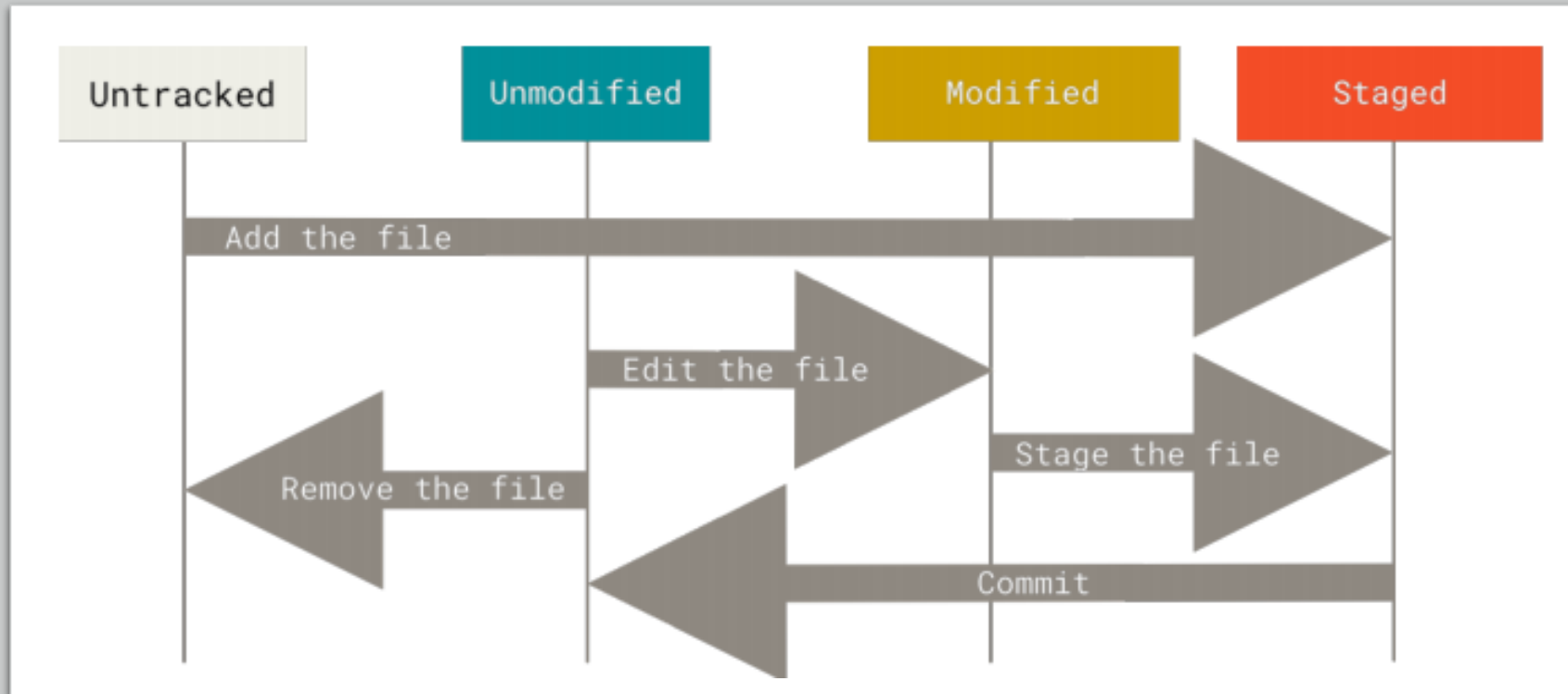
***git clone*** adalah perintah untuk menyalin repo git yang ada, mendapatkan semua data yang lengkap

**Perintah** git clone <url>

Membuat direktori libgit2, yang ada pada .git, memeriksa Salinan apakah cocok dengan latest version, semua salinan ada

**Dapat juga dengan nama direktori sendiri**

**Dapat juga menggunakan protokol transfer berbeda**



## Record Perubahan Directory

- Membuat perubahan dan ingin melakukan snapshots
- Status: **tracked** file yang diketahui oleh git, yang ada snapshot terakhir tentang modified, staged, dsb. **Untracked** adalah yang tidak ada snapshot terakhir dan tidak ada dalam area tagging

# Check Status Files

- **git status** digunakan untuk cek status file

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

*Menunjukkan bahwa direktori kerja bersih artinya tidak ada file yang berstatus tracked*

- Saat membuat file dari awal..

```
$ echo 'My Project' > README
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    README

nothing added to commit but untracked files present (use "git add" to track)
```

Berstatus untracked file karena tidak ada snapshot terakhir/sebelumnya



# Tracking File Baru

- **git add** digunakan untuk melacak file baru

```
$ git add README
```

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)

    new file:   README
```

Kemudian berubah menjadi staged, dan changes to be committed, dan snapshots terakhir berubah

## Mengganti File Staging

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   README

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   CONTRIBUTING.md
```

- **git add** lebih cocok dengan menambahkan konten ini ke commit berikutnya

***File tracked telah dimodifikasi namun belum berstatus staged, perlu perintah git add agar staged***

```
$ git add CONTRIBUTING.md
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
```

```
    new file:   README
   modified:   CONTRIBUTING.md
```

```
$ vim CONTRIBUTING.md
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   README
   modified:   CONTRIBUTING.md

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

   modified:   CONTRIBUTING.md
```

```
$ git add CONTRIBUTING.md
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
```

```
    new file:   README
   modified:   CONTRIBUTING.md
```

# Short Status

- **git status -s** untuk menampilkan status dengan informasi yang pendek

```
$ git status -s
M README
MM Rakefile
A lib/git.rb
M lib/simplegit.rb
?? LICENSE.txt
```

- A mewakili file yang baru ditambahkan di staging area, M mewakili file yang di Modified
- **Kiri** menandakan status pada staging area dan **kanan** status working tree
- Yang MM atau dua menandakan bahwa sudah diModify, Di Stage, namun di modify lagi, sehingga mempunyai 2 status staged dan unstaged

# Ignore Files Git

- Berfungsi untuk membuat class dari suatu file yang tidak ingin Git untuk otomatis ditambahkan atau untacked file tersebut

```
$ cat .gitignore  
*.[oa]  
*~
```

- \* untuk mengabaikan

**Ketentuan lain untuk .git ignore adalah**

1. Garis kosong atau # akan diabaikan
2. Standar glob diterapkan secara rekursif di semua working tree
3. / diawal untuk menghindari keterulangan dan / diakhir untuk menentukan direktori
4. ! Untuk meniadakan pola

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

```
*.a
```

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

```
!lib.a
```

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

```
/TODO
```

```
# ignore all files in any directory named build
```

```
build/
```

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

```
doc/*.txt
```

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

```
doc/**/*.pdf
```

## Melihat Perubahan Status Staging Area

```
$ git diff --staged
diff --git a/README b/README
new file mode 100644
index 0000000..03902a1
--- /dev/null
+++ b/README
@@ -0,0 +1 @@
+My Project
```

```
$ git diff
diff --git a/CONTRIBUTING.md b/CONTRIBUTING.md
index 8ebb991..643e24f 100644
--- a/CONTRIBUTING.md
+++ b/CONTRIBUTING.md
@@ -65,7 +65,8 @@ branch directly, things can get messy.
 Please include a nice description of your changes when you submit your PR;
 if we have to read the whole diff to figure out why you're contributing
 in the first place, you're less likely to get feedback and have your change
-merged in.
+merged in. Also, split your changes into comprehensive chunks if your patch is
+longer than a dozen lines.

If you are starting to work on a particular area, feel free to submit a PR
that highlights your work in progress (and note in the PR title that it's
```


- **git diff** digunakan untuk mengetahui status perubahan dengan detail
- **git diff --staged** untuk mengetahui apa yang telah di staging area jika ingin di commit setelahnya

## Commit Perubahan

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Your branch is up-to-date with 'origin/master'.
#
# Changes to be committed:
#   new file:   README
#   modified:   CONTRIBUTING.md
#
~
~
~
".git/COMMIT_EDITMSG" 9L, 283C
```

- Modified harus ditambahkan **git add** untuk dapat ke staging area
- **git commit** digunakan untuk commit terhadap perubahan
- Hasil dari commit seperti ketika git status sebelumnya



- 
- Master = branch, (463d) = checksum untuk menandai, berapa perubahan, statistic tentang baris dan perubahan
  - Commit merekam semua snapshot di area staging. Sebelum di mofied bisa menambahkan yang lain untuk di commit kedalam history

```
$ git commit -m "Story 182: fix benchmarks for speed"
[master 463dc4f] Story 182: fix benchmarks for speed
2 files changed, 2 insertions(+)
create mode 100644 README
```

# Skipping Staging Area

- **git commit -a** untuk skip pada area staging sehingga tidak harus menjalankan git add

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   CONTRIBUTING.md

no changes added to commit (use "git add" and/or "git commit -a")
$ git commit -a -m 'Add new benchmarks'
[master 83e38c7] Add new benchmarks
1 file changed, 5 insertions(+), 0 deletions(-)
```

## Removing Files

- **git rm** digunakan untuk menghapus file dari **kumpulan tracked file** dan **tidak ada juga pada pada list untracked file** alias bersih

```
$ git rm PROJECTS.md
rm 'PROJECTS.md'
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    deleted:    PROJECTS.md
```

Jika ingin menyimpan pada hard drive dulu atau total tidak dihapus adalah dengan cara

```
$ git rm --cached README
```

Atau menghapus secara keseluruhan dengan:

```
$ git rm \**
```

# Moving Files

```
$ git mv file_from file_to
```

```
$ git mv README.md README
```

```
$ git status
```

On branch master

Your branch is up-to-date with 'origin/master'.

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

```
renamed:    README.md -> README
```

A dark blue, irregular ink splash or blotch serves as the background for the text. The splash has a textured, painterly appearance with various shades of blue and some lighter, misty edges. It is centered on a white background.

# Viewing Commit History

- **git log** adalah cara paling mudah untuk melihat history
- **git log -p -2** untuk menampilkan history dengan limit
- **git log -stat** untuk menampilkan history yang lebih singkat
- **git log --pretty=oneline** untuk menampilkan yang lebih rapi
- **git log --pretty=format:%..** Untuk menampilkan rapi dengan format

|     |   |
|-----|---|
| %H  | Commit hash                                     |
| %h  | Abbreviated commit hash                         |
| %T  | Tree hash                                       |
| %t  | Abbreviated tree hash                           |
| %P  | Parent hashes                                   |
| %p  | Abbreviated parent hashes                       |
| %an | Author name                                     |
| %ae | Author email                                    |
| %ad | Author date (format respects the --date=option) |

# Perbedaan antara **Author** dan **Committer**

- Author adalah yang pertama memulai proyek tersebut
- Sedangkan commiter adalah orang yang paling terakhir mengaplikasikan proyek tersebut
- Jika telah Bersama-sama mengirimkan patch pada suatu proyek maka dua-duanya mendapatkan kredit



# Limitting Log Output

- Sangat perlu digunakan karena defaultnya git log menampilkan pipe pada seluruh pager
- **Pembatasan waktu (--since dan --until)**

```
$ git log --since=2.weeks
```

- **Atau berdasarkan author dengan (--author)**
- **Atau dengan menggunakan string (-S ...)**

```
$ git log -S function_name
```

|                                |  |
|--------------------------------|--|
| <code>-&lt;n&gt;</code>        | Show only the last n commits   |
| <code>--since, --after</code>  | Limit the commits to those made after the specified date.                    |
| <code>--until, --before</code> | Limit the commits to those made before the specified date.                   |
| <code>--author</code>          | Only show commits in which the author entry matches the specified string.    |
| <code>--committer</code>       | Only show commits in which the committer entry matches the specified string. |
| <code>--grep</code>            | Only show commits with a commit message containing the string                |
| <code>-S</code>                | Only show commits adding or removing code matching the string                |

# Undoing Things

---

# Undoing Commit

```
$ git commit --amend
```

```
$ git commit -m 'Initial commit'  
$ git add forgotten_file  
$ git commit --amend
```

# Unstaging Staged File

```
$ git add *  
$ git status  
On branch master  
Changes to be committed:  
  (use "git reset HEAD <file>..." to unstage)
```

```
renamed:    README.md -> README  
modified:   CONTRIBUTING.md
```

```
$ git reset HEAD CONTRIBUTING.md  
Unstaged changes after reset:  
M   CONTRIBUTING.md  
$ git status  
On branch master  
Changes to be committed:  
  (use "git reset HEAD <file>..." to unstage)  
  
renamed:    README.md -> README  
  
Changes not staged for commit:  
  (use "git add <file>..." to update what will be committed)  
  (use "git checkout -- <file>..." to discard changes in working directory)  
  
modified:   CONTRIBUTING.md
```

# Unmodifying Modified File

```
$ git checkout -- CONTRIBUTING.md
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    renamed:    README.md -> README
```

Hampir semua bisa dipulihkan

# Working With Remotes

- Terkadang membutuhkan kolaborasi dengan tim ketika mengelola proyek git
- Sehingga harus tahu bagaimana melakukannya secara remote melalui internet atau global repo
- Mencakup segala hal yang biasa dilakukan di local namun ini terjadi secara global atau remote melalui internet

# Melihat Remote

- **git remote** digunakan untuk dapat mengkoneksikan dengan remote
- **origin** artinya yang anda ingin clon repo untuk dapat di remote dalam repo anda

```
$ git remote -v  
origin https://github.com/schacon/ticgit (fetch)  
origin https://github.com/schacon/ticgit (push)
```

- Bisa remote banyak sehingga dapat pull dari banyak juga

```
$ cd grit  
$ git remote -v  
bakkdoor https://github.com/bakkdoor/grit (fetch)  
bakkdoor https://github.com/bakkdoor/grit (push)  
cho45 https://github.com/cho45/grit (fetch)  
cho45 https://github.com/cho45/grit (push)
```



## Menambahkan Remote Repositories

- **git remote add** untuk menambahkan remote repositori setelah dilakukan cloning sebelumnya
- **git fetch pb** digunakan untuk menambahkan repositori yang **pb** sendiri merupakan string pengganti url

```
$ git remote
origin
$ git remote add pb https://github.com/paulboone/ticgit
$ git remote -v
origin  https://github.com/schacon/ticgit (fetch)
origin  https://github.com/schacon/ticgit (push)
pb      https://github.com/paulboone/ticgit (fetch)
pb      https://github.com/paulboone/ticgit (push)
```

```
$ git fetch pb
remote: Counting objects: 43, done.
remote: Compressing objects: 100% (36/36), done.
remote: Total 43 (delta 10), reused 31 (delta 5)
Unpacking objects: 100% (43/43), done.
From https://github.com/paulboone/ticgit
* [new branch]      master    -> pb/master
* [new branch]      ticgit    -> pb/ticgit
```

# Fetching dan Pulling dari Remote Repositories

- Perintah **git fetch** hanya mengunduh data dari remote ke *repository local saja* tidak otomatis merge ke pekerjaan anda saat ini.
- Harus merging secara manual

# Pushing Your Remotes

- Jika ingin membagikan ke banyak orang maka harus di **push** terlebih dahulu dengan perintah **git push <remote> <branch>**
- Jika ingin push master ke origin server dapat menggunakan **git push origin master**
- **Tetapi...**
- Hanya bekerja jika clon dari server yang anda dapat access write dan oranglain tidak push pada waktu yang bersamaan. Jika bersamaan push maka akan direject. Solusinya anda harus fetch projeknya terlebih dahulu dan dapat push lagi jika diizinkan

# Inspect Remote

- **Git remote show <remote>** adalah perintah yang digunakan untuk mengetahui informasi tentang remote termasuk tracking branch info

```
$ git remote show origin
* remote origin
Fetch URL: https://github.com/schacon/tiegit
Push URL: https://github.com/schacon/tiegit
HEAD branch: master
Remote branches:
  master                                tracked
  dev-branch                            tracked
Local branch configured for 'git pull':
  master merges with remote master
Local ref configured for 'git push':
  master pushes to master (up to date)
```

# Renaming and Removing Remotes

- **git remote rename** digunakan untuk mengganti nama remote

```
$ git remote rename pb paul  
$ git remote  
origin  
paul
```

- **git remote remove** digunakan untuk memindahkan remote repo yang mungkin tidak berkontribusi lagi atau memang ingin diganti

```
$ git remote remove paul  
$ git remote  
origin
```



TAGGING

# Listing Your Tags

- **git tag (with optional)** digunakan untuk menampilkan tag secara singkat
- Atau dapat juga menggunakan optional tertentu

```
$ git tag -l "v1.8.5*"
v1.8.5
v1.8.5-rc0
v1.8.5-rc1
v1.8.5-rc2
v1.8.5-rc3
v1.8.5.1
v1.8.5.2
```

# Creating Tags

- **Git Support 2 tipe tags yaitu lightweight dan announted**
  - a. **Lighweight** adalah tag yang mirip branch yang tidak berubah – hanya pointer untuk menunjukan commit tertentu
  - b. **Annotated tags** adalah tag yang disimpan sepenuhnya di Database Git, berupa checksumed yang berisi contain tagger name. email. Dan tanggal maupun tag message. **Biasanya digunakan** untuk membuat tag yang harus berisi banyak info



# Annotated Tags

- Membuat annotated tag sangat mudah. Hanya dengan menambahkan `-a` ketika run tag seperti berikut

```
$ git tag -a v1.4 -m "my version 1.4"
$ git tag
v0.1
v1.3
v1.4
```

- `-m` digunakan sebagai message dari tag yang Bersama disimpan dengan tag tersebut
- Tag tersebut serta message dapat ditampilkan dengan perintah **git show <tag>**

# Lightweight Tags

- Yang berisi dari checksum yang disimpan dalam file, tidak ada info lain yang disimpan. Dibuat tidak dengan menggunakan tag option karena memang yang disimpan adalah tag sederhana **git tag**

```
$ git tag v1.4-lw
$ git tag
v0.1
v1.3
v1.4
```

```
$ git show v1.4-lw
commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    Change version number
```

# Tagging Later

- Dapat juga tag setelah commit dengan cara menambahkan commit checksum dibelakangnya seperti berikut:

```
$ git log --pretty=oneline
15027957951b64cf874c3557a0f3547bd83b3ff6 Merge branch 'experiment'
a6b4c97498bd301d84096da251c98a07c7723e65 Create write support
0d52aaab4479697da7686c15f77a3d64d9165190 One more thing
6d52a271eda8725415634dd79daabbc4d9b6008e Merge branch 'experiment'
0b7434d86859cc7b8c3d5e1dddfed66ff742fcbc Add commit function
4682c3261057305bdd616e23b64b0857d832627b Add todo file
```

```
$ git tag -a v1.2 9fceb02
```

# Sharing Tags

- Defaultnya perintah git push tidak transfer tags ke remote server harus ditambahkan manual ketika di remote server, prosesnya sama seperti sharing remote branches **git push origin <tagname>**
- **Atau** jika ingin banyak tags yang ingin di push pada satu waktu yaitu dengan **git push origin ==tags**

# Deleting Tags

- Untuk menghapus pada local repository dapat menggunakan **git tag -d <tagname>** dengan catatan itu tidak di remove tag dari banyak remote server
- Ada dua cara untuk delete tag dari sebuah remote server
  - a. Menggunakan

The first variation is `git push <remote> :refs/tags/<tagname>`:

```
$ git push origin :refs/tags/v1.4-lw
To /git@github.com:schacon/simplegit.git
- [deleted]          v1.4-lw
```

- b. Atau dengan **git push origin -delete <tagname>**

# Checking Out Tags

- Jika ingin melihat versi dari files yang menunjuk ke tag, dapat menggunakan perintah **git checkout**

```
$ git checkout 2.0.0
Note: checking out '2.0.0'.
```

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using `-b` with the checkout command again. Example:

```
git checkout -b <new-branch>
```

```
HEAD is now at 99ada87... Merge pull request #89 from schacon/appendix-final
```

```
$ git checkout 2.0-beta-0.1
Previous HEAD position was 99ada87... Merge pull request #89 from schacon/appendix-final
HEAD is now at df3f601... Add atlas.json and cover image
```

```
$ git config --global alias.co checkout  
$ git config --global alias.br branch  
$ git config --global alias.ci commit  
$ git config --global alias.st status
```

# Git Aliases

- Jika tidak ingin mengetikkan perintah seluruh teks dapat dengan mudah dengan mengatur alias dengan perintah

# KESIMPULAN

- Pada pertemuan kali ini kita telah mempelajari basic operasi dari Git





TERIMA KASIH