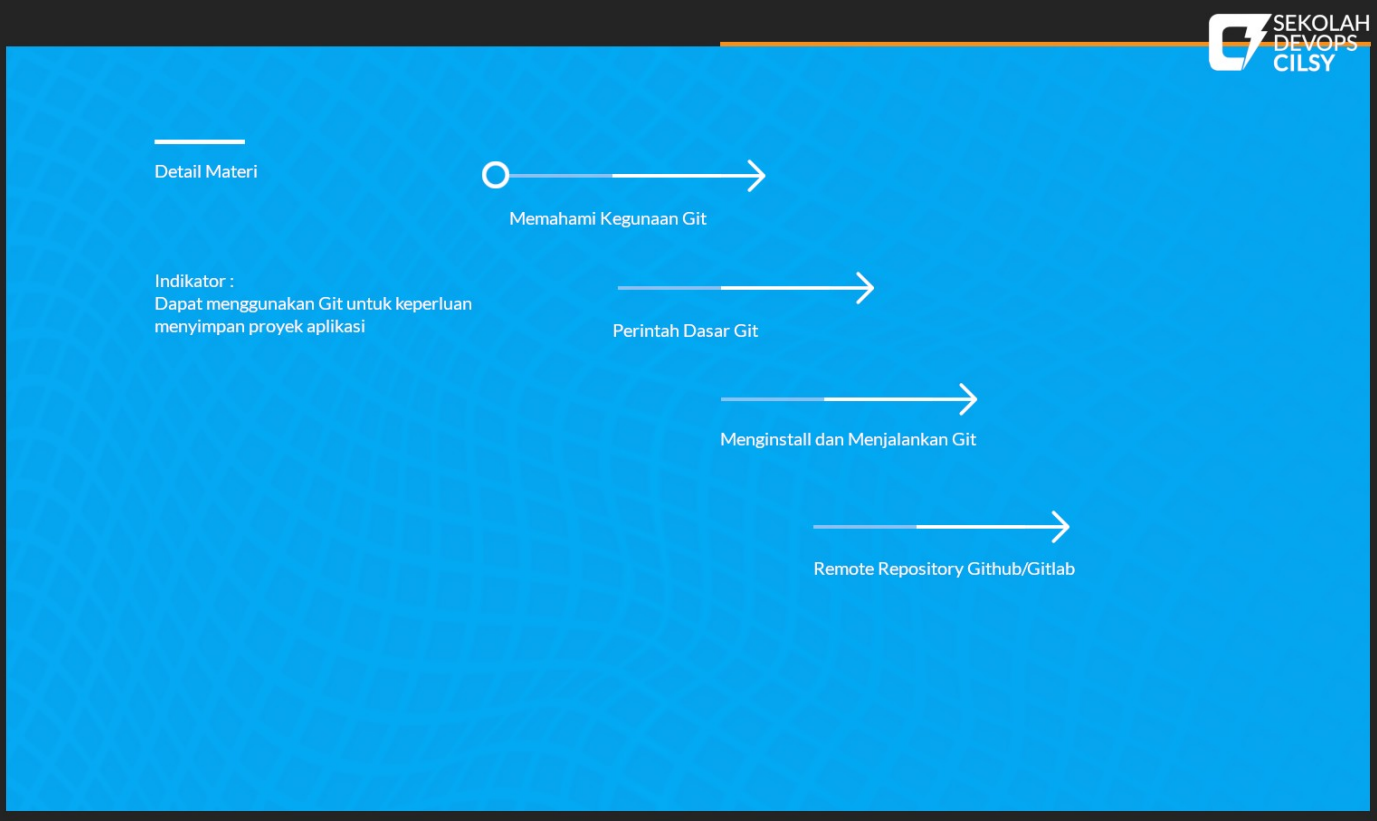
Bab 4

Pengenalan Git



Modul Sekolah DevOps Cilsy

Penulis : Adi Saputra, Irfan Herfiandana & Tresna Widiyaman
Editor : Muhammad Fakhri Abdillah, Rizal Rahman & Tresna Widiyaman
Revisi Batch 4

Penerbit : **PT. Cilsy Fiolution Indonesia**
Web Site : https://cilsyfiolution.com , https://devops.cilsy.id

Daftar Isi

4.
Pengenalan Git part 2: Studi Kasus

Learning Outcomes

Setelah selesai mempelajari bab ini, peserta mampu :

1. Mampu melakukan kolaborasi pada Git

Outline Materi

1. Membuat Repository
2. Pushing Data
3. Pulling Data
4. Working with Branches
5. Merging
6. Cherry-pick
7. Penggunaan Tags

4.1. Membuat Repository

Pada modul sebelumnya, kita sudah mempelajari bagaimana cara membuat local git repository. Sekarang, kita akan mempelajari cara berkolaborasi dan juga best practice Git. Dimulai dengan membuat server repository yang digunakan untuk menyimpan dan memanage kode-kode yang kita punya.

Seperti yang sudah kita ketahui, Git dapat menggunakan 4 protokol untuk melakukan transport data:

- Local
- Secure Shell (SSH)
- Git
- HTTPS

Kita akan membahasnya keempat metode tersebut di bagian selanjutnya. Untuk saat ini, kita akan mencoba membuat repositori baru. Lakukan perintah-perintah berikut

```
Ajo@server:~$ mkdir webproject
#Create the folder
Ajo@server:~$ cd webproject
#go inside it
Ajo@server:~/webproject$ git init --bare
Initialized empty Git repository in /home/ajo/webproject
```

Dengan perintah diatas, kita sudah membuat direktori webproject dan menginisialisasi git bare repository.

4.1.1. Local

Local repository adalah protocol yang paling dasar (basic), karena remote repository disini adalah local directory. Protokol ini dapat digunakan apabila member memiliki akses ke remote repository, contohnya melalui NFS.

Pada kasus ini, programmer dapat melakukan clone repository di jaringan local, dengan menggunakan perintah berikut:

```
Ajo@server:~$ git clone /opt/git/webproject.git
```

Mari kita ambil sebuah contoh. Jimmy adalah seorang programmer yang telah menulis code line. Jimmy telah menginisialisasi local git repository di dalam directory dan telah men-set remote location untuk bare repository menggunakan perintah berikut

```
Jimmy@local:~/webproject$ git init
Jimmy@local:~/webproject$ git remote add origin /opt/git/webproject.git
```

Terdapat kelebihan dan kekurangan dalam penggunaan local repository ini, diantaranya:

Kelebihan:

- Mudah untuk dishare ke member lain
- Akses ke repo cepat

Kekurangan:

- Sulit untuk di setup pada sebuah shared network
- Akses cepat hanya jika file accessnya cepat

4.1.2. SSH

Secure shell (SSH) adalah protokol yang paling sering digunakan, terutama apabila server repository ada di tempat lain.

Untuk mengakses/meng-clone repository, programmer harus melakukan perintah berikut:

```
Ajo@local:~$ git clone ssh://username@server/webproject.git
```

Untuk menggunakan protokol SSH, programmer perlu menginstall SSH key pada remote repository untuk melakukan push dan pull ke repository tersebut.

Kita ambil contoh Jimmy diatas, apabila menggunakan protokol SSH, maka perintah yang digunakan adalah sebagai berikut.

```
Jimmy@local:~/webproject$ git init

Jimmy@local:~/webproject$ git remote add origin
ssh://username@server/webproject.git
```

4.1.3. Git

Git transport mirip dengan SSH, tetapi tanpa menggunakan keamanan apapun. Kita tidak dapat melakukan push data secara default, tetapi kita dapat mengaktifkan fitur ini. Hal ini sama sekali bukan ide yang bagus, karena siapapun yang menemukan alamat repositori dapat melakukan push data. Seperti dalam studi kaksu diatas, programmer harus melakukan clone secara lokal dengan menggunakan perintah berikut:

```
Ajo@local:~$ git clone git://username@server/webproject.git
```

Pada kasus Jimmy, kita bisa menggunakan perintah sebagai berikut.

```
Jimmy@local:~/webproject$ git init

Jimmy@local:~/webproject$ git remote add origin
git://username@server/webproject.git
```

4.1.4. HTTPS

Protokol HTTPS adalah protokol yang paling mudah disetup. Siapapun yang mempunyai akses ke webserver dapat melakukan clone data.

Programmer dapat menggunakan perintah berikut untuk mengclone data.

```
Ajo@local:~$ git clone https://server/webproject.git
```

Untuk kasus Jimmy, kita dapat menggunakan perintah sebagai berikut:

```
Jimmy@local:~/webproject$ git init .

Jimmy@local:~/webproject$ git remote add origin http://server/webproject.git
```

4.2. Pushing Data on Remote Repositories

Setelah Jimmy melakukan inisialisasi git repository baru pada directory local, ketika ia selesai melakukan coding, dan ia menambahkan remote repository menggunakan protokol SSH, dia melakukan commit dan melakukan push code yang telah ditulisnya. Langkah yang ia lakukan adalah sebagai berikut:

```
Jimmy@local:~/webproject$ git add .
Jimmy@local:~/webproject$ git commit -m 'add my code'
[master (commit racine) 83fcc8a] add my code
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 index.html
create mode 100644 readme.txt
Jimmy@local:~/webproject$ git push –u origin master
Counting objects: 3, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 225 bytes | 0 byte/s, done.
Total 3 (delta 0), reused 0 (delta 0)
```

Sekarang, repository tersebut memiliki 2 file baru yaitu index.html dan readme.txt.

4.3. Pulling data from the Repository

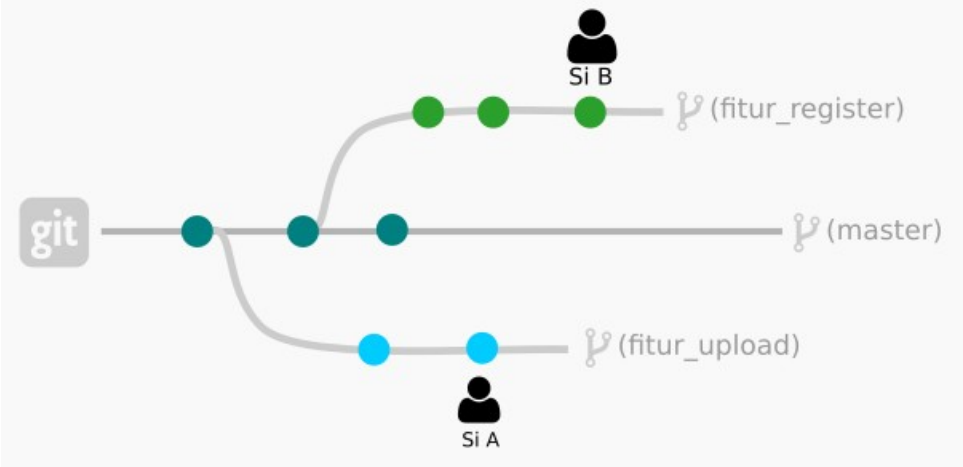
Dalam studi kasus ini, ada programmer yang akan melakukan pull data untuk mendapatkan file baru. Perintah yang digunakan untuk melakukan pull data adalah sebagai berikut:

```
Ajo@local:~/webproject$ git pull origin master
```

Perintah ini akan mengecheck dan membandingkan local commit hash kita dengan remote hash. Apabila pada remote adalah versi terakhir dari data yang disimpan (latest), maka git akan mencoba untuk melakukan merge data yang ada pada master branch yang ada di local. Perintah ini sama halnya dengan mengeksekusi *git fetch* (get remote data) dan *git merge*.

4.4. Working with Branches

Git memungkinkan kita untuk membuat branch (cabang). Kita dapat bekerja dalam branch yang berbeda dengan programmer lain. Hal ini dapat digunakan untuk melakukan pemisahan fungsi, fitur, dan/atau pemisahan code supaya lebih teratur. Selain itu, dengan adanya branch akan mencegah terjadinya conflict antar programmer.



Ilustrasi Branch

Pada modul ini, kita akan mencoba memberikan studi kasus bagaimana cara membuat branch dan nantinya akan ada kolaborasi programmer lainnya. Contohnya, Jimmy akan membuat branch bernama test. Maka perintah yang digunakan adalah sebagai berikut.

```
Jimmy@local:~/webproject$ git branch test
```

Untuk mulai menggunakan branch tersebut, kita perlu melakukan checkout ke branch yang akan dituju. Perintah yang digunakan oleh Jimmy adalah sebagai berikut.

```
Jimmy@local:~/webproject$ git checkout test
#do some changes inside the readme.txt files
Jimmy@local:~/webproject$ git commit -a -m 'edit readme'
Jimmy@local:~/webproject$ git checkout master
```

Selanjutnya, kita akan mencoba beberapa skenario pada branch ini. Contohnya, kita akan merename branch test dengan perintah berikut.

```
Jimmy@local:~/webproject$ git branch -m <old_name> <new_name>
```

Contohnya;

```
Jimmy@local:~/webproject$ git branch -m test testbaru
```

Lalu, kita bisa mendelete branch dengan perintah berikut:

```
Jimmy@local:~/webproject$ git branch -d testbaru
```

Kita akan menemui error apabia ada perubahan yang belum di commit. Kita dapat melakukan force dengan perintah berikut:

```
Jimmy@local:~/webproject$ git branch -D testbaru
```

Kemudian kita akan membuat branch test lagi, dengan perintah

```
Jimmy@local:~/webproject$ git branch test
```

Lalu kita push sebuah file dengan perintah

```
Jimmy@local:~/webproject$ git push origin test
```

Selanjutnya, untuk mengetahui perbedaan antar branch, kita bisa gunakan perintah berikut

```
Jimmy@local:~/webproject$ git diff master test
```

Apabila Jimmy ingin menghapus branch dari remote repository, maka ia dapat menggunakan perintah berikut.

```
Jimmy@local:~/webproject$ git branch -d origin/test
```

4.5. Merging

Git memungkinkan kita untuk mengkombinasikan perubahan pada dua branch yang berbeda. Hal ini disebut dengan *merging*.

Perintah yang diguanakan adalah git merge. Kita dapat melakukan merge dari suatu branch ke current branch. Contoh penggunaannya adalah sebagai berikut.

```
Ajo@local:~/webproject$ git checkout master
Ajo@local:~/webproject$ git merge test
```

Hal ini akan mengakibatkan perubahan yang terjadi pada branch test akan digabungkan dengan file yang ada pada branch master.

4.6. Cherry-pick

Perintah ini memungkinkan kita memilih commit dari branch untuk menerapkannya ke branch lain. Patch ini akan dikenali sebagai commit baru pada branch yang telah dipilih. Untuk lebih mengerti perintah git cherrypick ini, kita akan belajar menggunakan studi kasus dibawah.

Jimmy membuat branch baru dengan nama jim dari branch master, dan ia menambahkan file di branch tersebut.

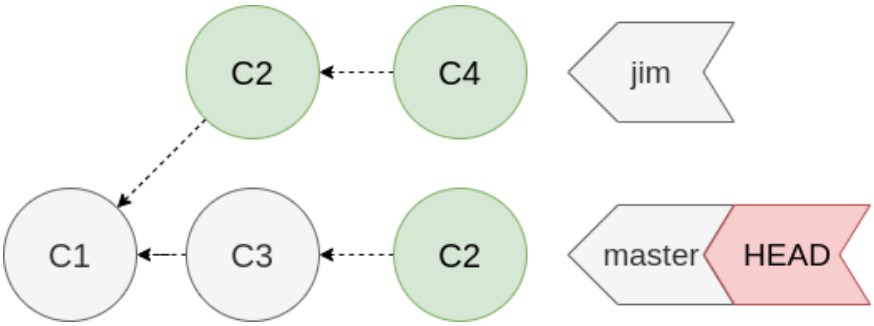
```
Jimmy@local:~/webproject$ git checkout -b jim
Jimmy@local:~/webproject$ touch home.html
Jimmy@local:~/webproject$ git add home.html
Jimmy@local:~/webproject$ git commit -m 'add homepage'
Jimmy@local:~/webproject$ echo "<html> … </html>" > home.html
Jimmy@local:~/webproject$ git commit -m 'add content inside home'
```

Seperti yang kita lihat dari perintah-perintah diatas, bahwa Jimmy membuat file home.html, menambahkannya ke git, dan melakukan commit. Selanjutnya ia mengedit file tersebut dan melakukan commit lagi. Sekarang kita lihat log dari commit history pada branch ini.

```
Jimmy@local:~/webproject$ git log --oneline
4f6ec45 add content inside home
22c45b7 add homepage
```

Sekarang, Jimmy akan menerapkan commit pertama pada master branch.

```
Jimmy@local:~/webproject$ git checkout master
Jimmy@local:~/webproject$ git cherry-pick 22c45b7
```



Maka hasilnya adalah commit pertama yang dilakukan pada branch jim akan diterapkan pada branch master. Sederhananya, apabila kita visualisasikan, ilustrasinya akan seperti pada gambar dibawah

Sekarang, anggaplah terdapat kesalahan pada cherry-pick yang dilakukan, dan Jimmy ingin membatalkannya. Maka perintah yang digunakan adalah sebagai berikut.

```
Jimmy@local:~/webproject$ git cherry-pick --abort
```

Nah, apabila Anda ingin melakukan rollback dari cherry-pick yang dilakukan, maka Anda memiliki 2 pilihan perintah, yaitu:

- Apabila ada di private branch, maka kita dapat menggunakan perintah *git rebase*.
- Apabila ada di public branch, maka kita dapat menggunakan perintah *git revert*.

4.7. Penggunaan Tags

Git memiliki option untuk memberikan tag pada commit yang dilakukan pada sebuah repository, sehingga kita akan lebih mudah untuk mengidentifikasi versi dari commit yang dilakukan. Umumnya, tag digunakan pada sebuah versi aplikasi yang di commit.

Untuk membuat tag pada git, kita bisa menggunakan perintah berikut:

```
Jimmy@local:~/webproject$ git tag 1.0.0
#Memberikan tag dengan menggunakan description
Jimmy@local:~/webproject$ git tag 1.0.0 -m 'Release 1.0.0'
#Menggunakan Commit
Jimmy@local:~/webproject$ git tag 1.0.0 -m 'Release 1.0.0' commit_hash
```

Tag juga dapat dihapus, tapi secara default, ini hanya terjadi di local repository. Apabila kita ingin melakukan push pada tag yang dihapus, maka kita perlu menspesifikkannya. Caranya, kita list semua tag yang tersedia.

```
Jimmy@local:~/webproject$ git tag
0.1.0
0.1.5
0.2.0
0.9.0
1.0.0
```

Selanjutnya, kita hapus tag terakhir.

```
Jimmy@local:~/webproject$ git tag -d 1.0.0
#Push it remotely
Jimmy@local:~/webproject$ git push origin tag 1.0.0
```

4.8. Exercise

Buatlah kelompok dengan jumlah anggota 2 orang. Lakukanlah operasi sebagai berikut:

1. Buatlah sebuah repository bernama script-otomasi.
2. Masing-masing anggota membuat branch dengan namanya sendiri.
3. Orang pertama membuat script otomasi untuk menginstall Apache web server, sementara orang kedua membuat script otomasi untuk menginstall MySQL server pada masing-masing branch.
4. Lakukan merging dari 2 script tersebut. Apakah hal tersebut bisa dilakukan? Apabila tidak bisa (atau muncul pesan error), mengapa hal tersebut terjadi? Lalu, apa yang harus dilakukan apabila kita ingin membuat sebuah script master yang berfungsi untuk melakukan instalasi Apache web server dan MySQL server dalam sekali run? Jelaskan!