



Gitlab

Principles of Gitlab, Operation in Git, and Features of Gitlab

Daftar Isi

A. Konsep Version Control	3
B. Git	3
C. Cara Kerja Git	5
D. Istilah dalam Git	6
E. Instalasi dan Konfigurasi Awal Git	8
F. Git Commands	14
G. Gitlab	15
H. Fitur Gitlab	16
References	21

A. Konsep Version Control

Version control adalah sebuah sistem yang merekam perubahan-perubahan dari sebuah berkas atau sekumpulan berkas dari waktu ke waktu sehingga kita dapat menilik kembali versi khusus suatu saat nanti. Sebagai contoh, pada buku ini kita akan menggunakan sumber kode perangkat lunak sebagai berkas-berkas yang direkam dengan version control, walau pada kenyataannya kita dapat melakukan ini dengan hampir semua jenis berkas pada komputer.

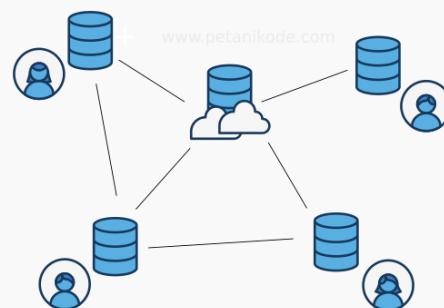
Jika kita adalah seorang perancang grafis atau web dan ingin menyimpan setiap versi dari sebuah gambar atau layout (yang tentunya kita ingin melakukannya), sebuah Version Control System (VCS) adalah hal yang bijak untuk digunakan. VCS memperbolehkan kita untuk mengembalikan berkas-berkas ke keadaan sebelumnya, mengembalikan seluruh proyek kembali ke keadaan sebelumnya, membandingkan perubahan-perubahan di setiap waktu, melihat siapa yang terakhir mengubah sesuatu yang mungkin menimbulkan masalah, siapa dan kapan yang mengenalkan sebuah isu dan banyak lagi. Menggunakan VCS secara umum juga berarti bahwa jika kita melakukan kesalahan atau menghilangkan berkas, kita dapat dengan mudah memulihkannya. Sebagai tambahan, kita mendapatkan semua ini dengan biaya yang sangat sedikit.

B. Git

Git adalah salah satu sistem pengontrol versi (Version Control System) pada proyek perangkat lunak yang diciptakan oleh Linus Torvalds. Pengontrol versi

bertugas mencatat setiap perubahan pada file proyek yang dikerjakan oleh banyak orang maupun sendiri. Git dikenal juga dengan distributed revision control (VCS terdistribusi), artinya penyimpanan database Git tidak hanya berada dalam satu tempat saja.

Semua orang yang terlibat dalam pengkodean proyek akan menyimpan database Git, sehingga akan memudahkan dalam mengelola proyek baik online maupun offline. Dalam Git terdapat merge untuk menyebut aktivitas penggabungan kode. Sedangkan pada VCS (Version Control System) yang terpusat... database disimpan dalam satu tempat dan setiap perubahan disimpan ke sana.



VCS terpusat memiliki beberapa kekurangan:

- Semua tim harus terkoneksi ke jaringan untuk mengakses source-code;
- Tersimpan di satu tempat, nanti kalau server bermasalah bagaimana?

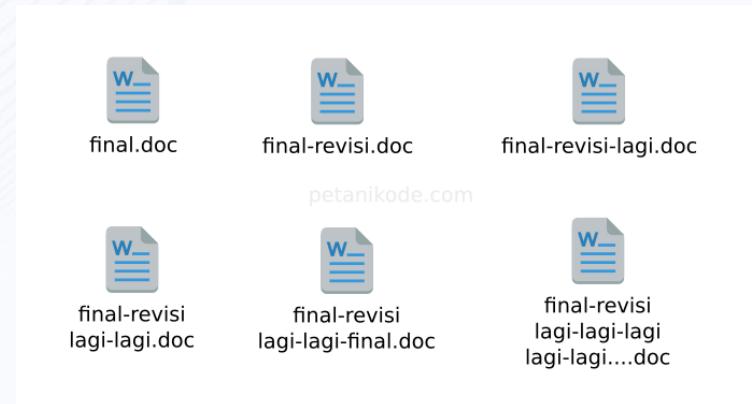
Karena itu, Git hadir untuk menutupi kercurangan yang dimiliki oleh VCS terpusat.

Selain untuk mengontrol versi, git juga digunakan untuk kolaborasi. Saat ini Git menjadi salah satu tool terpopuler yang digunakan pada pengembangan software open souce maupun closed source. Google, Microsoft, Facebook dan berbagai perusahaan raksasa lainnya menggunakan Git.

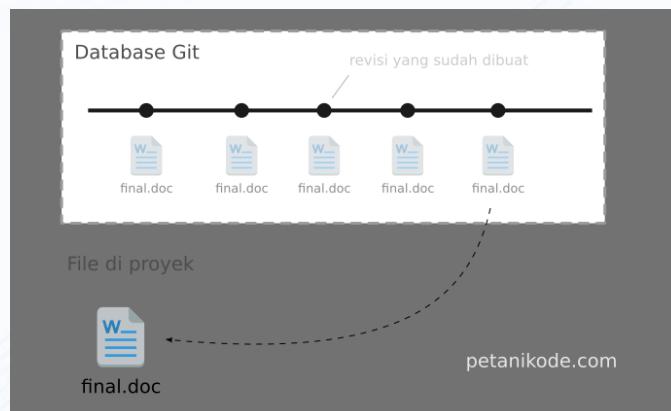
C. Cara Kerja Git

Git sebenarnya akan memantau semua perubahan yang terjadi pada file proyek. Lalu menyimpannya ke dalam database.

Sebelum menggunakan Git:



Setelah menggunakan Git:



Apa perbedaannya?

Saat kita ingin menyimpan semua perubahan pada file, biasanya kita membuat file baru dengan "save as". Lalu, file akan menumpuk dalam direktori proyek seperti pada ilustrasi di atas.

Tapi setelah menggunakan Git, hanya akan ada satu file dalam proyek dan perubahannya disimpan dalam database. Git hanya akan menyimpan delta

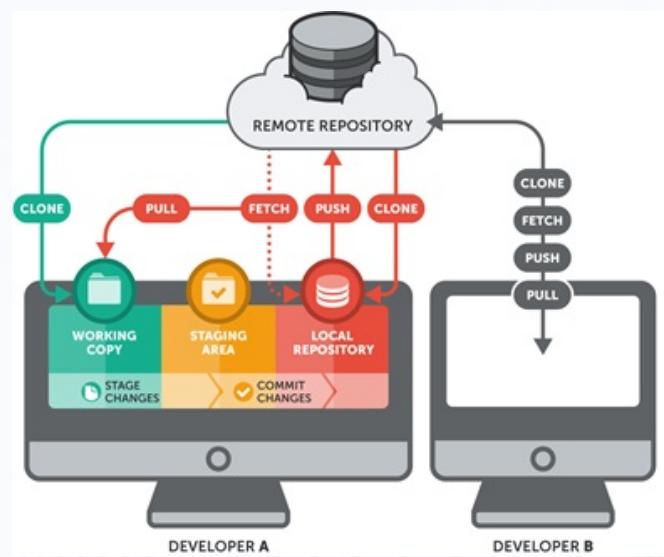
perubahannya saja, dia tidak akan menyimpan seluruh isi file yang akan memakan banyak memori. Git memungkinkan kita kembali ke versi revisi yang kita inginkan.

D. Istilah dalam Git

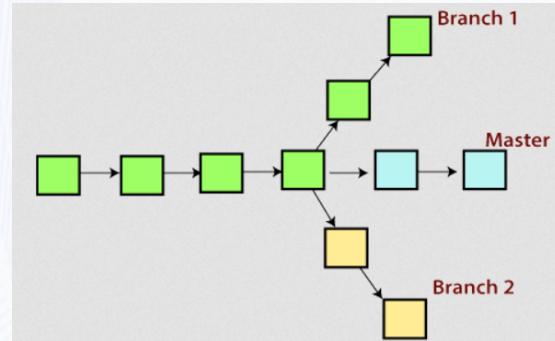
- Repository : Istilah yang digunakan untuk direktori / folder project yang menggunakan Git
- Local Repository: Repository yang tersimpan pada komputer lokal (offline)
- Remote Repository : Repository yang tersimpan pada layanan git cloud (online)
- Branch : Cabang dari repository yang dapat digunakan untuk membuat perubahan. Branch merupakan serangkaian commit yang berkaitan sehingga kalau digambar seperti garis lurus berisi banyak commit. Satu repository bisa berisi banyak branch.
- Snapshot : potret kondisi file dan folder pada saat tertentu
- Commit : snapshot yang disimpan di repository
- Master : nama branch default yang diberikan git pada waktu kita membuat repository. Branch master ini tidak istimewa. Dia bisa dihapus dan di-rename sesuka hati.
- Head : ujung branch, commit terbaru di dalam branch
- HEAD : head yang sedang aktif. Walaupun satu repository bisa memiliki banyak branch, tapi cuma satu yang aktif.
- Working folder : folder berisi file dan folder tempat kita bekerja. Biasanya working folder berisi banyak file source code untuk aplikasi yang sedang

kita buat. Git memantau working folder ini, dan bisa mengetahui file dan folder mana yang sudah berbeda dari posisi commit terakhir. Perbedaan atau perubahan ini bisa disimpan menjadi commit baru, atau dikembalikan ke kondisi sebelum diubah.

- Staging area : snapshot dari working folder yang akan kita simpan pada saat commit. Ini adalah fitur unik Git yang tidak dimiliki version control lain. Dengan adanya staging area, kita bisa memilih perubahan mana yang akan di-commit dan mana yang tidak.
- Object store : ini adalah database tempat semua commit disimpan.



Gambar di atas menggambarkan ilustrasi mengenai remote repository dan local repository. Remote repository tersimpan di layanan / platform git cloud (internet / online). Para developer dapat mengambil repository tersebut ke komputer masing-masing dan melakukan perubahan (menambah fitur, menghapus kode, dll). Repository yang berada di setiap komputer developer dinamakan local repository.



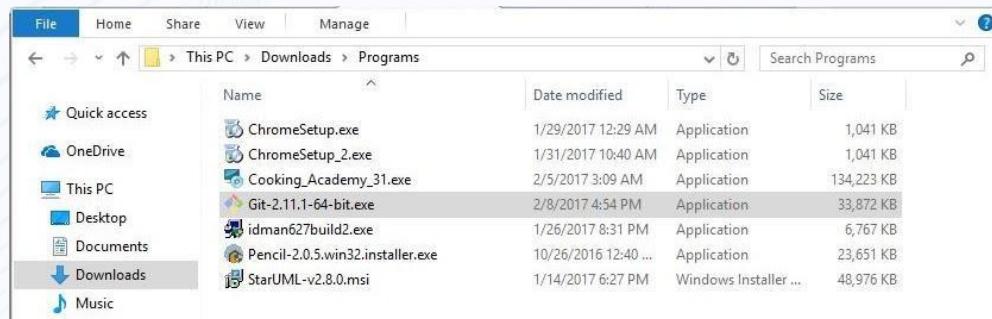
Gambar di atas menggambarkan ilustrasi mengenai branch. Setiap repository mempunyai main branch yang bernama master. Pada gambar tersebut terdapat branch lain yaitu branch 1 dan branch 2 yang merupakan cabang dari branch master. Branching sangat berguna untuk melakukan develop fitur atau remove fitur tanpa mengganggu kode yang terdapat pada branch master. Jika fitur sudah selesai didevelop maka branch fitur tersebut dapat digabung ke dalam branch master.

E. Instalasi dan Konfigurasi Awal Git

Instalasi Git di Windows memang tidak seperti di Linux yang menjalankan com langsung terinstal. Silahkan buka website resminya Git (git-scm.com). Kemudian unduh Git sesuai dengan arsitektur komputer kita. Kalau menggunakan 64bit, unduh yang 64bit. Begitu juga kalau menggunakan 32bit.

Berikut ini merupakan langkah-langkah Install Git di Windows.

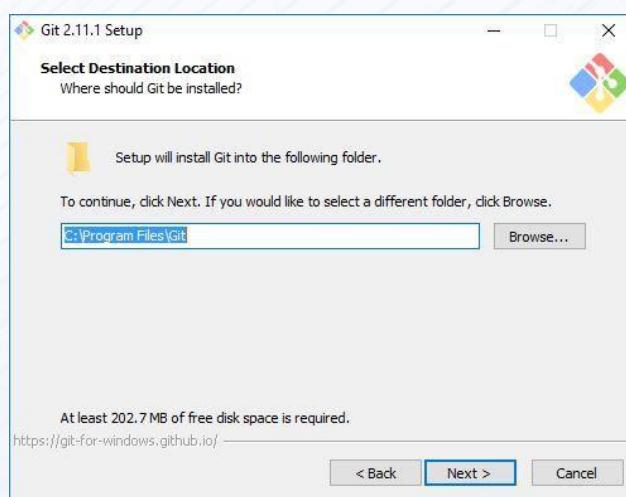
- Silahkan klik 2x file installer Git yang sudah diunduh.



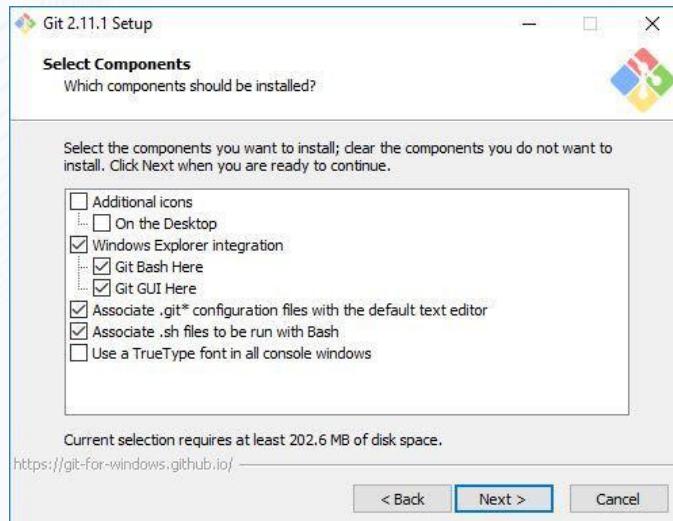
- Maka akan muncul informasi lisensi Git, klik Next > untuk melanjutkan.



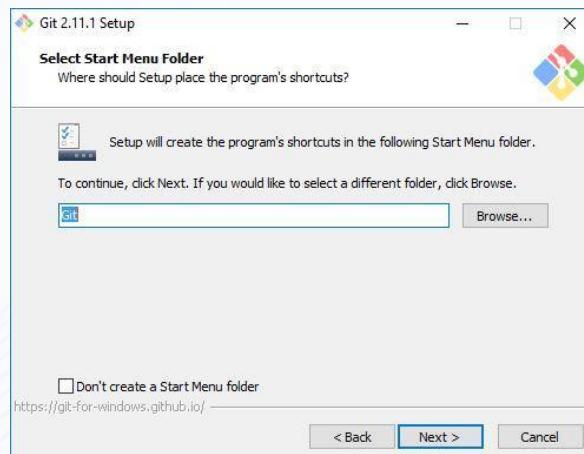
- Selanjutnya menentukan lokasi instalasi. Biarkan saja apa adanya, kemudian klik Next >.



4. Selanjutnya pemilihan komponen, biarkan saja seperti ini kemudian klik Next >.



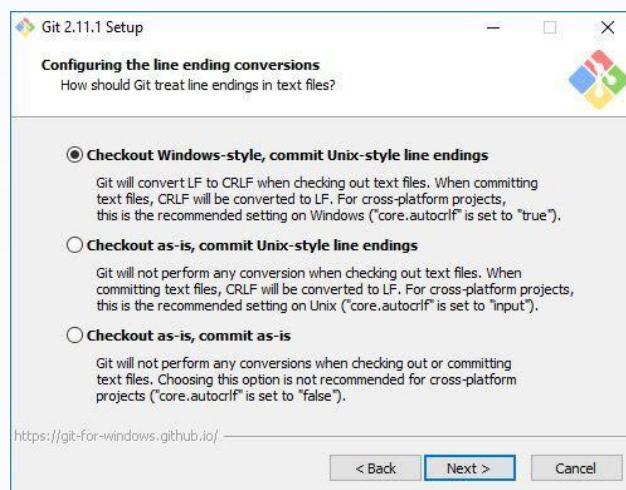
5. Selanjutnya pemilihan direktori start menu, klik Next >.



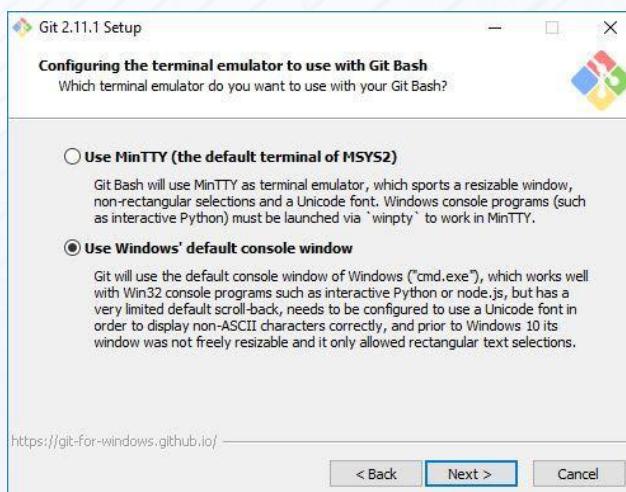
6. Selanjutnya pengaturan PATH Environment. Pilih yang tengah agar perintah git dapat dikenali di Command Prompt (CMD). Setelah itu klik Next >.



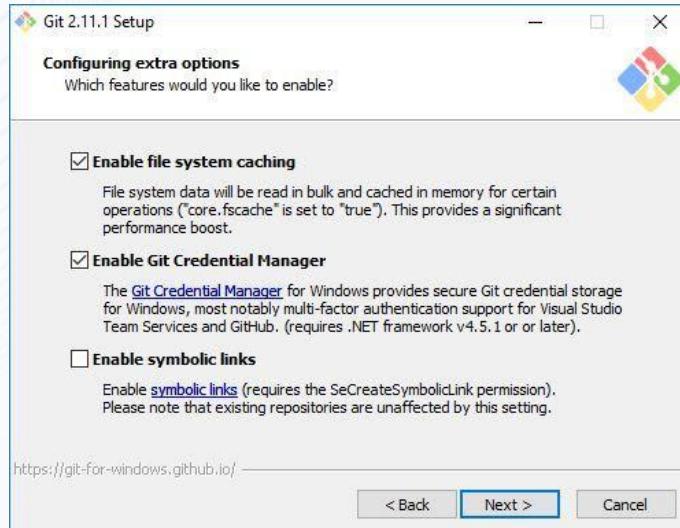
7. Selanjutnya konversi line ending. Biarkan saja seperti ini, kemudian klik Next >.



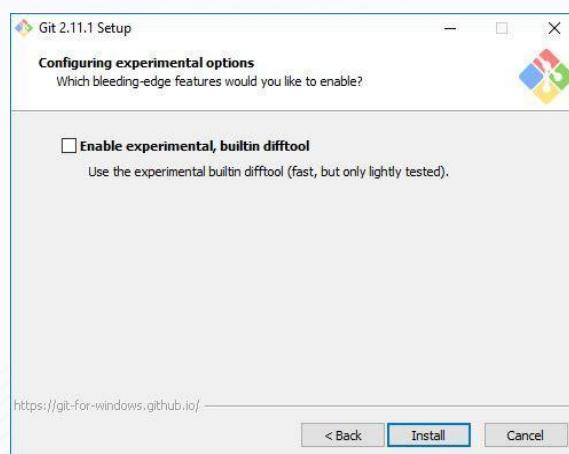
8. Selanjutnya pemilihan emulator terminal. Pilih saja yang bawah, kemudian klik Next >.



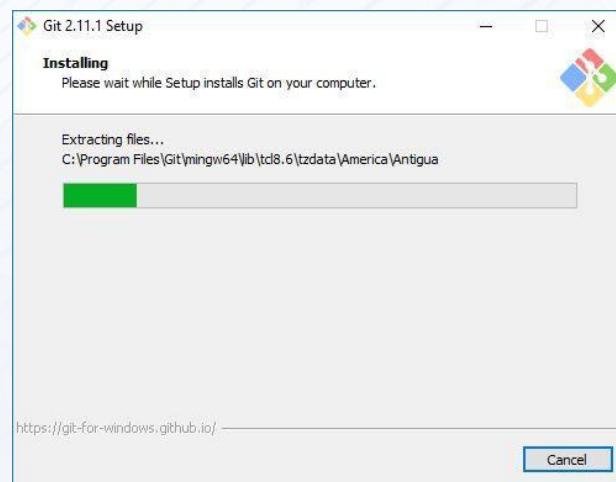
9. Selanjutnya pemilihan opsi ekstra. Klik saja Next >.



10. Selanjutnya pemilihan opsi eksperimental, langsung saja klik Install untuk memulai instalasi.



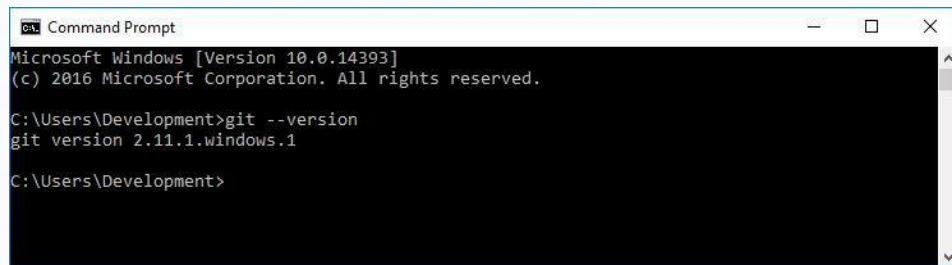
11. Tunggu beberapa saat, instalasi sedang dilakukan.



12. Setelah selesai, kita bisa langsung klik Finish.



13. Selamat, Git sudah terinstal di Windows. Untuk mencobanya, silahkan buka CMD atau PowerShell, kemudian ketik perintah git --version.



Konfigurasi Awal yang Harus Dilakukan

Ada beberapa konfigurasi yang harus dipersiapkan sebelum memulai menggunakan Git, seperti name dan email.

1. Silahkan lakukan konfigurasi dengan perintah berikut ini.

```
git config --global user.name "Petani Kode"  
git config --global user.email contoh@petanikode.com
```

2. Kemudian periksa konfigurasinya dengan perintah:

```
git config --list
```

3. Apabila berhasil tampil seperti gambar berikut ini, berarti konfigurasi berhasil.



```
Welcome to fish, the friendly interactive sh
dian@petanikode~> git config --list
user.email=dian@petanikode.com
user.name=Dian
dian@petanikode~>
```

Konfigurasi core.editor bersifat opsional. Sedangkan name dan email wajib. Jika kita memiliki akun Github, Gitlab, Bitbucket atau yang lainnya maka username dan email harus mengikuti akun tersebut agar mudah diintegrasikan. Selain konfigurasi awal ini, kita juga bisa konfigurasi SSH key untuk Github, Gitlab, dan Bitbucket.

F. Git Commands

Berikut ini merupakan beberapa command dasar yang sering digunakan pada Git :

- **git config** : untuk konfigurasi yang harus dipersiapkan sebelum memulai menggunakan Git, seperti name dan email. Diset saat baru saja menginstall git pada komputer.
- **git init** : untuk pembuatan repository. Nantinya perintah ini akan membuat sebuah direktori bernama .git di dalam proyek kita. Direktori ini digunakan Git sebagai database untuk menyimpan perubahan yang kita lakukan.
- **git clone <path>** : untuk menduplikat repository yang ada (repository yang berada di remote)
- **git status** : untuk mengecek file mana yang berada di branch mana
- **git add <file_name>** : untuk mulai melacak file baru. File tersebut ditambahkan ke dalam git project lokal komputer

- **git commit** : Commit perubahan dalam project. Sekarang area staging diatur sesuai keinginan kita, kita dapat melakukan commit. Ingatlah bahwa apa pun yang masih belum dalam area staging- file apa pun yang telah kita buat atau modifikasi yang belum kita jalankan git add on sejak kita mengeditnya – tidak akan masuk ke komit ini. Mereka akan tetap sebagai file yang dimodifikasi pada disk komputer (lokal). Dalam hal ini, katakanlah terakhir kali kita menjalankan git status, kita melihat bahwa semuanya telah di area staging, jadi kita siap untuk melakukan perubahan.
- **git remote** : Untuk melihat server remote mana yang telah kita konfigurasikan. Ini mencantumkan nama pendek dari setiap remote handler yang kita tentukan.
- **git checkout <branch_name>** : Untuk berpindah dari branch yang aktif dengan branch yang sudah dipilih
- **git push** : Untuk digunakan dalam mengirimkan perubahan file yang dilakukan setelah di commit ke remote repository
- **git pull** : Untuk menarik atau mengambil source code dari repository
- **git merge <branch_name>** : untuk menggabungkan branch aktif serta branch yang dipilih

G. Gitlab

GitLab merupakan salah satu git repository berbasis cloud sekaligus DevOps platform. Platform ini membantu developer untuk memonitor, melakukan tes dan deploy kode mereka. GitLab sendiri memiliki beberapa fungsi yang berguna untuk meringankan task dari para developer

- **Manage** – mengelola proyek dan melihat bagaimana performa bisnis perusahaan Anda.
- **Plan** – GitLab menyediakan tool planner agar tim Anda dapat tersinkronisasi.
- **Create** – Branching tool memungkinkan Anda membuat, melihat dan mengelola kode.
- **Verify** – Build-in CI/CD untuk tes terotomasi dan juga reporting.
- **Package** – membuat supply chain perangkat lunak yang dapat diandalkan dan terkontrol.
- **Secure** – GitLab menyediakan fitur yang membuat aplikasi Anda aman dengan software license compliance.
- **Release** – CD yang terintegrasi memungkinkan Anda untuk mengantarkan kode secara otomatis.
- **Configure** – GitLab memungkinkan Anda untuk mengkonfigurasi aplikasi dan infrastruktur.
- **Monitor** – Mengurangi kemungkinan error dan insiden yang terjadi dengan fitur monitoring.
- **Protect** – Aplikasi dan infrastruktur Anda akan dijamin keamanannya dengan fitur keamanan GitLab.

H. Fitur Gitlab

GitLab telah menyediakan layanan mereka secara gratis. Namun untuk menggunakan fiturnya secara keseluruhan, kita mungkin akan membutuhkan paket yang premiumnya. Mereka menyediakan paket Free (gratis), Premium dan juga Ultimate. Paket Premium GitLab ini ditujukan

untuk koordinasi dalam skala kecil atau tim. Paket ini bisa didapatkan dengan harga 19 USD per user/bulan. Sedangkan untuk paket Ultimatnya, GitLab membanderolnya dengan harga 99 USD per user/bulan.

Secara mendasar mengubah cara tim Pengembangan, Keamanan, dan Operasi berkolaborasi dan membangun perangkat lunak - GitLab menyediakan semua alat DevOps penting dalam satu platform DevOps. Dari ide hingga produksi, GitLab membantu tim meningkatkan waktu siklus dari minggu ke menit, mengurangi biaya pengembangan, mempercepat waktu pemasaran, dan menghadirkan aplikasi yang lebih aman dan sesuai.

Berikut ini merupakan fitur yang disediakan oleh Gitlab :

1. Plan

Terlepas dari proses tim, GitLab menyediakan alat perencanaan yang andal agar semua orang tetap tersinkronisasi. GitLab memungkinkan perencanaan dan manajemen portofolio melalui epik, grup (program), dan pencapaian untuk mengatur dan melacak kemajuan. Terlepas dari metodologi Anda dari Waterfall hingga DevOps, pendekatan GitLab yang sederhana dan fleksibel untuk perencanaan memenuhi kebutuhan tim kecil hingga perusahaan besar. GitLab membantu tim mengatur, merencanakan, menyelaraskan, dan melacak pekerjaan proyek untuk memastikan tim bekerja pada hal yang benar pada waktu yang tepat dan mempertahankan visibilitas ujung ke ujung dan ketertelusuran masalah selama siklus hidup pengiriman dari ide hingga produksi.

2. Create

Buat, lihat, dan kelola kode dan data proyek melalui branching tools yang andal. GitLab memungkinkan perencanaan dan manajemen portofolio melalui epik, grup (program), dan pencapaian untuk mengatur dan

melacak kemajuan. Terlepas dari metodologi Anda dari Waterfall hingga DevOps, pendekatan GitLab yang sederhana dan fleksibel untuk perencanaan memenuhi kebutuhan tim kecil hingga perusahaan besar. GitLab membantu tim mengatur, merencanakan, menyelaraskan, dan melacak pekerjaan proyek untuk memastikan tim bekerja pada hal yang benar pada waktu yang tepat dan mempertahankan visibilitas ujung ke ujung dan ketertelusuran masalah selama siklus hidup pengiriman dari ide hingga produksi.

3. Verify

Pertahankan standar kualitas yang ketat untuk kode produksi dengan pengujian dan pelaporan otomatis. GitLab membantu tim pengiriman sepenuhnya merangkul integrasi berkelanjutan untuk mengotomatiskan build, integrasi, dan verifikasi kode mereka. Kemampuan CI terdepan di industri GitLab memungkinkan pengujian otomatis, Pengujian Keamanan Analisis Statis, pengujian Keamanan Analisis Dinamis, dan analisis kualitas kode untuk memberikan umpan balik cepat kepada pengembang dan penguji tentang kualitas kode mereka. Dengan pipeline yang memungkinkan pengujian bersamaan dan eksekusi paralel, tim dengan cepat mendapatkan wawasan tentang setiap komitmen, yang memungkinkan mereka mengirimkan kode berkualitas lebih tinggi dengan lebih cepat.

4. Package

Buat supply chain software yang konsisten dan dapat diandalkan dengan manajemen paket bawaan. GitLab memungkinkan tim untuk mengemas aplikasi dan dependensi mereka, mengelola kontainer, dan membuat artefak dengan mudah. Private, secure, container, dan package registry

sudah terpasang dan telah dikonfigurasi sebelumnya untuk bekerja secara mulus dengan manajemen kode sumber GitLab dan pipeline CI/CD. Pastikan akselerasi DevOps dan waktu yang lebih cepat untuk memasarkan dengan saluran perangkat lunak otomatis yang mengalir bebas tanpa gangguan.

5. Secure

Kemampuan keamanan, terintegrasi ke dalam siklus pengembangan Anda. GitLab menyediakan Pengujian Keamanan Aplikasi Statis (SAST), Pengujian Keamanan Aplikasi Dinamis (DAST), Pemindaian Kontainer, dan Pemindaian Ketergantungan untuk membantu Anda memberikan aplikasi yang aman bersama dengan kepatuhan lisensi.

6. Release

Solusi CD terintegrasi GitLab memungkinkan Anda mengirim kode tanpa sentuhan, baik di satu atau seribu server. GitLab membantu mengotomatiskan rilis dan pengiriman aplikasi, memperpendek siklus hidup pengiriman, merampingkan proses manual, dan mempercepat kecepatan tim. Dengan Continuous Delivery (CD) zero-touch yang terpasang langsung ke dalam pipeline, penerapan dapat diotomatisasi ke berbagai lingkungan seperti pementasan dan produksi, dan sistem hanya mengetahui apa yang harus dilakukan tanpa diberi tahu - bahkan untuk pola yang lebih canggih seperti penerapan canary. Dengan tanda fitur, audit/penelusuran bawaan, dan lingkungan sesuai permintaan, Anda akan dapat memberikan lebih cepat dan lebih percaya diri daripada sebelumnya.

7. Configure

Konfigurasikan aplikasi dan infrastruktur Anda. GitLab membantu tim untuk mengonfigurasi dan mengelola lingkungan aplikasi mereka. Integrasi yang kuat ke Kubernetes mengurangi upaya yang diperlukan untuk mendefinisikan dan mengonfigurasi infrastruktur yang diperlukan untuk mendukung aplikasi Anda. Lindungi akses ke detail konfigurasi infrastruktur utama seperti kata sandi dan informasi login dengan menggunakan 'variabel rahasia' untuk membatasi akses hanya ke pengguna dan proses yang sah.

8. Monitor

Membantu mengurangi tingkat keparahan dan frekuensi insiden. Dapatkan umpan balik dan alat untuk membantu Anda mengurangi keparahan dan frekuensi insiden sehingga Anda dapat sering merilis perangkat lunak dengan percaya diri.

9. Govern

Kelola kerentanan keamanan, kebijakan, dan kepatuhan di seluruh organisasi Anda. GitLab membantu pengguna mengelola kerentanan keamanan, kebijakan, dan kepatuhan di seluruh organisasi mereka.

References

<https://git-scm.com/book/id/v2/>

<https://www.petanikode.com/git-untuk-pemula/>

<https://github.com/endymuhardin/buku-git/blob/master/buku/03-penggunaan.md>

<https://www.goldenfast.net/blog/gitlab-vs-github/>

<https://about.gitlab.com/features/>