**SECTION 6 – Versioning Control with Git**

Video KMQE – Version Control and Branch Management (Git) (1:26:51)

Outline

1. What is Versioning?

2. Git Install

3. Setting Up (Git Init, Clone, Config)

4. Saving Changes (Git Status, Add, Commit, Diff, Stash, Ignore)

5. Inspecting Repository and Undoing (Git Log, Checkout, Reset)

6. Syncing (Git Remote, Fetch, Push, Pull)

7. Branches

8. Pull Request

9. Workflow Collaboration

1. **What is Versioning?**

Mengatur versi dari source code program. Melakukan sebuah pengaturan versi atau pelacakan perubahan dari setiap code program kita. Contoh ketika skripsi membuat sebuah dokumen proposal, bimbingan, revisi dll (pada saat revisi tidak langsung diedit karena kita ingin track isi sebelumnya perubahannya apa, maka kita buat file baru) contoh: (Revisi - Revisi Fix – Revisi Fix Final – Revisi Fix Final Last). Untuk apa kita perlu mengetahui perubahan sebelumnya? Untuk tracking kesalahan sebelumnya dan siapa yang mengubah, apa saja yang diubah dll.

Management project untuk mengetahui setiap perubahan apa yang dilakukan.

“Revision Is A Must, Don’t Expect Every Code is Perfect” (Tidak ada code yang sempurna sekelas Google ataupun Apple) akan selalu ada hal yang di optimasi agar lebih cepat lagi dll. Akan ada sebuah update contoh: Perubahan versi android dll, Perubahan F5 pada Facebook dll.

Untuk mengecheck revisi banyak tools yang digunakan diantaranya:

* Version Control System (VCS)

Single User

(SCCS – 1972 Unix only)

(RCS – 1982 Cross Platform, text only)

Sebelumnya untuk memanagement sebuh tugas itu bersfiat localize karena internet masih belum se-power sekarang, dulu hanya berbasis komputer manual allan turning, masih belum menyambung antara satu komputer dengan komputer lain.

Centralized

(CVS – 1986 File focus)

(Perforce – 1995)

(Subversion – 2000 – track director structure)

(Microsoft Team Foundation Server – 2005)

Naik level dari localize mebjadi centralized sudah ada sistem terpusat, efek negatifnya sangat tergantung pada pusat.

Distributed

(Git – 2005) ada di local dan pusat server, local ter sync dengn pusat

(Mercurial – 2005)

(Bazaar – 2005)

Git (Salah satu version control system yang popular digunakan para developer untuk mengembangkan software secara bersama-sama), untuk tack perubahan yang berkolaborasi dengan developer lain yang sama-sama mengedit. Ada code review juga.

Real World Collaboration (Terdistribusi bukan tersentralisasi, artinya kita memiliki banyak komputer namun saling berhubungan, dengan menggunakan Git. Tapi tidak terpaku kepada server dalam Git. Cukup efektif untuk pengembangan). Dibuat oleh Linus Torvalds (2005) – Linux Kernel https://github.com/torvalds/linux.

Every Should Sync to The Remote Server (Pusat Server)

Remote Computer/Server (Pusat Sync)

My Computer John Computer Doe Computer

Jika mau localize ter-sync dengan Git, dibedakan menjadi dua file (Folders and File, History)

Git Repository (Folder Project)

Folders and Files History

Git akan men-track semua perubahan, siapa yang mengubah dll.

Git dapat meng-undo (mengembalikan) reset. Setiap perubahan ketika mau mengupdate ke server akan dikirimkan ke sebuah paket (Commit: berisi banyak file yang dapat di check perubahannya dibagian mana saja).

Commit (Riwayat Perubahan).

Yang paling popular pada Git (GitHub, GitBucket, dan GitLab) semacam sosial media-nya para developer. GitHub ada Public (Open Source dapat diakses oleh semua orang) dan Private(Bersifat Pribadi).

Membuat Repository dalam sebuah GitHub

1. Repositoryname (nama repository)

2. Description (deskripsi repo, bersifat opsional)

3. Access (Public & private)

4. Readme (Dokumentasi penting, terletak dibawah file project)

Gitignore (File besar), kalau file besar misal milik vendor pakai gitignore saja (berada di local) di enskripsi, meminimalisir keamanan

Choose a license (Open project, developer lain bisa ikut lihat namun ada aturan-aturan didalamnya)

5. Create Repository (Repository sudah dibuat)

Sync Git ke Local (Bisa memakai remote dan clone)

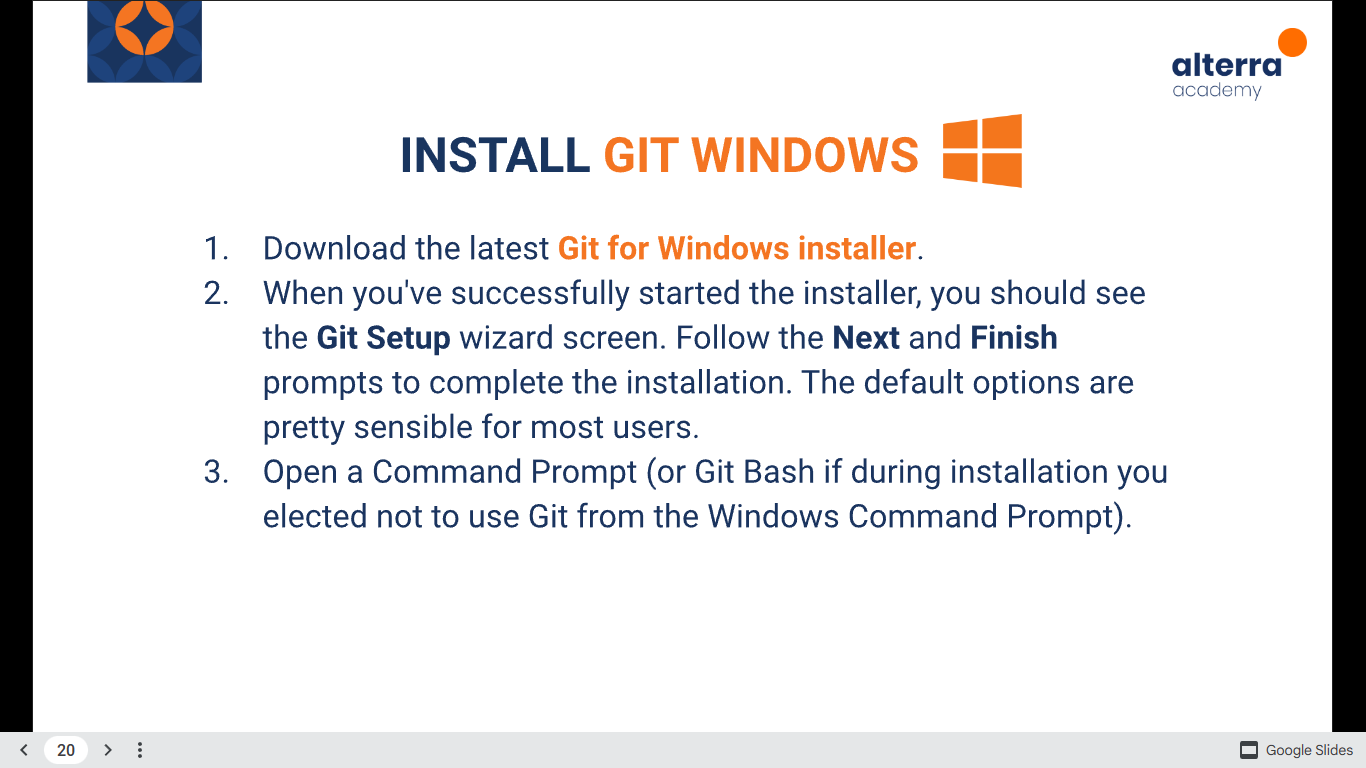
1. Project yang sudah ada tinggal dihubungkan (Clone: dipindah)

2. git remote add origin https://github.com/nisaauzz/QE-NISA’-UZZULFA.git

Ps: Di project besar membuat API wajib readme

* Source Code Manager (SCM)
* Revision Control System (RCS)

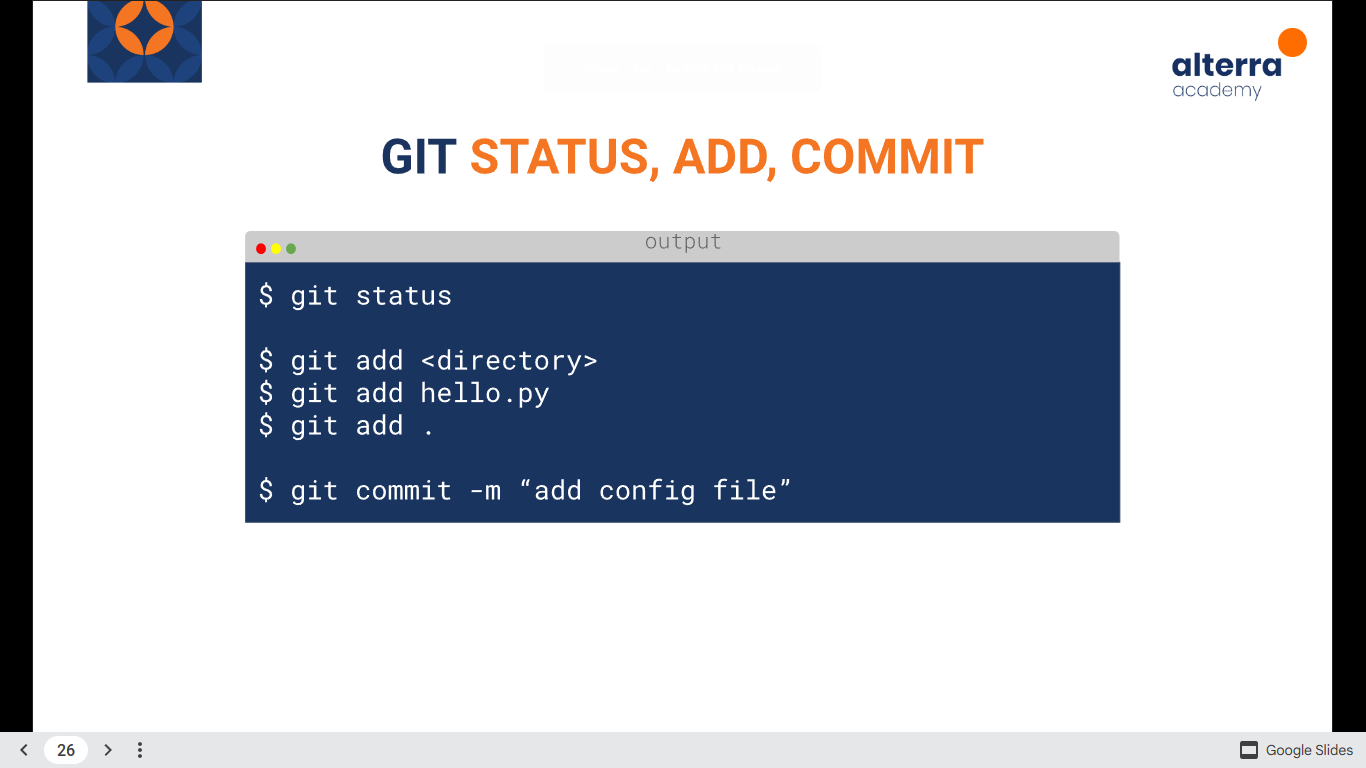
1. **Git Install**

****

1. **Setting Up (Git Init, Clone, Config)**

****

1. **Saving Changes (Git Status, Add, Commit, Diff, Stash, Ignore)**

****

****

The Staging Are (di local kita dibagi menjadi 3 bagian)

1. Working directory (git add)

2. Staging are (git comit: menambahi file-file yg dirubah)

3. Repository

Code:

git add main.go

git add .

git commit –m “inisialisasi project” (dari staging area dijadikam satu commit, jika di enter akan masuk keg it repository, sehingga dibalik layar ada satu commit yang ready untuk di sync ke server) (untuk mengirimkannya ke server menggunakan git push)

git push origin master(nama jalan)

git status (muncul nama branch dan perubahan yang ada di project, dan statusnya sudah di staging atau belum)

git add . (Mengubah dari working directory menuju staging area, bisa satu folder, atau langsung semua)

git commit –m “add config file (nama pesan yg ingin disampaikan)” hindari pesan yang tidak jelas

git status (mengecheck perubahan yg terjadi)

git diff (mengecheck perubahan apa saja yg terjadi di project kita)

git stash (sebuah penyimpanan sementara)

git stash apply

conflict (2 perubahan di 1 tempat yg sama)

add-commit-push

.gitignore (file untuk meng-eksekusi yang tidak dikirimkan ke sebuah github)

1. **Inspecting Repository and Undoing (Git Log, Checkout, Reset)**

****

****

Git Log (mengecheck perubahannya apa saja) git log –oneline

Setiap commit punya ID

CI/CD

Git Reset (melakukan pengembalian ke titik tertentu, pengembalian ini ada 2 jenis (--soft dan –hard), ketika mau kembali ke suatu titik, perubahan yg sudah terjadi akan tetep ke-track di staging area (--soft))

git reset code --soft

Checkout (pindah branch)

1. **Syncing (Git Remote, Fetch, Push, Pull)**

****

Push (mengirimkan commit ke server)

Fetch (mengambil data-data yang ada di dalam server)

git fetch

git fetch origin

1. **Branches**

****

Mengisolasi perubahan-perubahan apa saja yg ada di project

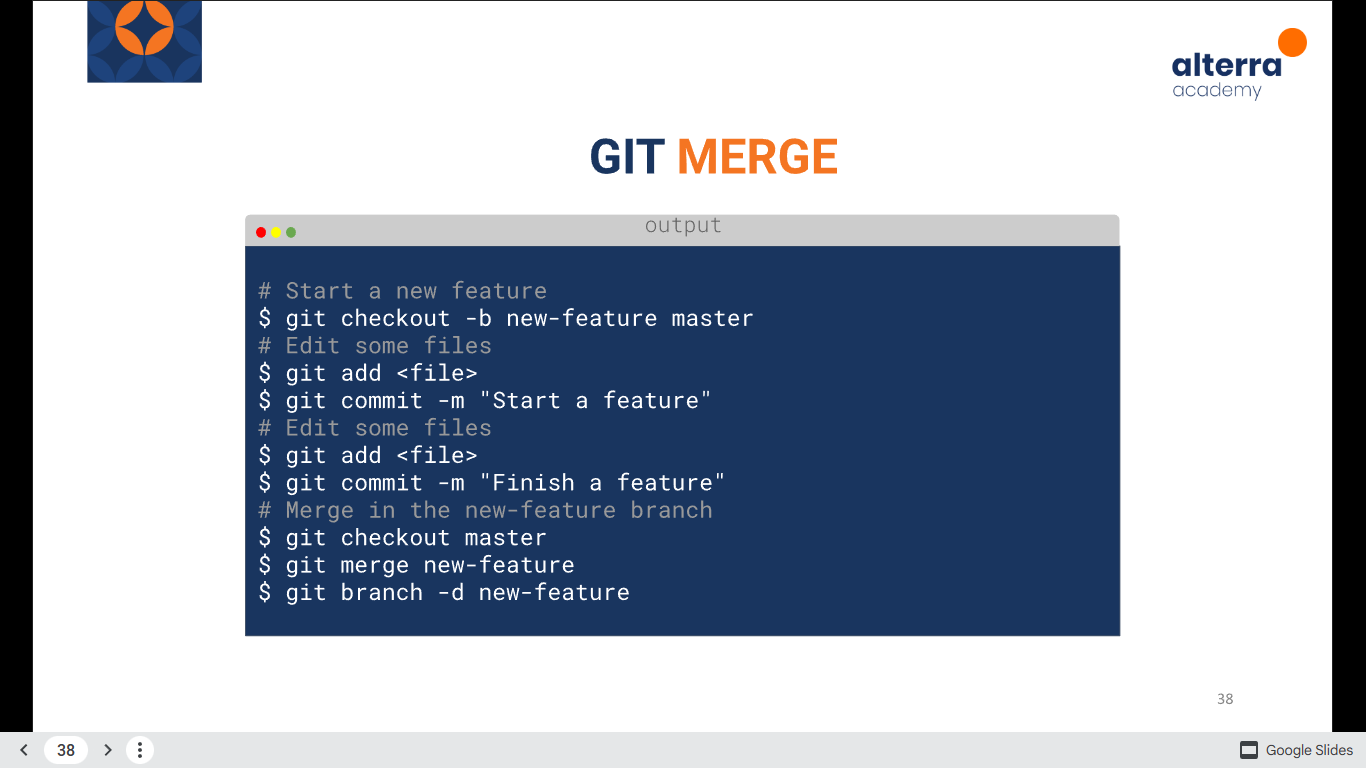
Contoh: misal ingin menambahkan fiture A

git branch (ingin melihat branch kita apa saja, yg berbintang yg sedang kita jalankan)

Git Merge (mengambil dari branch lain)

* Fast Forward (Graph nya jadi satu line)
* No Fast Forward (Graph nya bercabang, seolah-olah men-track pemisahan)

1. **Pull Request**

****

Merge ke master

1. **Workflow Collaboration (Graph)**

Branch nya ada apa saja, flow-nya bagaimana?

Alurnya, dari sini kita tahu bagaimana optimasinya.