Introdução ao Git e GitHub

GIT → Sistema de controle de versão distribuído. **GITHUB** → Repositório remoto (nuvem que hospeda o Git).

Navegação Básica no terminal

GUI → Graffic User Interface
CLI → Comand Line Interface

cmd → prompt de comando

Comandos do Windows:

- cd → change directory_navegar
- dir → directory_listar
- mkdir → make directory_criar
- del/rmdir → delete/remove directory_excluir
- cd .. → retrocede
- cls → clear screen_limpar
- TAB → completa automaticamente nomes de pastas/arquivos
- echo → retorna a informação digitada
- ">" → direcionamento de fluxo
- "↑" → navega entre comandos já utilizados no terminal

Instalação

Instalação Git

OBS: Marcar opção de Windows Explorer Integration (Git Bash Here)(Git GUI Here)

Tópicos fundamentais do Git

- SHA1 → Secure Hash Algorithm/Algorítmo único de encriptação/Conjunto de 40 dígitos de characteres
- Objetos Fundamentais:
  - Blobs → bloco que contém tipo/tamanho/conteúdo
  - Trees → armazenam blobs e outras trees_README/RAKEFILE/LIB
  - Commits → engloba tudo_inclusive mensagem de descrição/autor/horário

Autentificações Seguras do GitHub

- Chave SSH → temos a chave pública e privada/ O GitHub reconhece a máquina já configurada.
- Token → guardar o token em um arquivo do notebook, pois dps de gerado não temos mais acesso a ele _ usa o token no lugar da senha na autentificação para commit.

Ciclo de Vida do Arquivo no Git

- Untracked → Arquivos/pastas que o Git não tem ciência.
- Tracked → Arquivos/pastas monitorados pelo Git [para isso é precido add o arquivo ao Git (git add), e para voltar o arquivo ao untracked basta remover o mesmo do Git]

```
Há 3 estágios para os arquivos tracked
```

    - Unmodified → Arquivo adicionado ao Git e que não sofreu modificações OU Arquivo comitado (o arquivo após o commit volta ao estágio de unmodified)
    - Modified → Arquivo editado/modificado
    - Staged → Arquivo pronto para commit _ basta darmos um git add nos arquivos modificados

Ciclo

Unmodified → Modified → Staged → Commit → Unmodified

Commit

É uma documentação do meu código (salva, tem o autor, horário, mensagem...); é como se eu tirasse uma foto do meu código naquele momento.

Ambiente de desenvolvimento

Working Directory (criar/editar/trabalhar) | Staging Area (Staged) | Local Repository (arquivos comitados)

Do repositório Local empurramos para o repositório remoto

Servidor

Repositório remoto (GitHub)

Iniciando o Git

Comandos:

- ls → listar _ ls -a → listar mostrando pastas ocultas
- cd → navegar
- ctrl+l → limpar tela
- mkdir → criar pastas
- mv → mover pasta/arquivo
- git init → iniciar um repositório do git [dentro da pasta em questão]
- git status → mostra o status dos arquivos da pasta (se precisam ir pra staged/se precisam ser comitados...)
- git add → muda o estágio do arquivo/pasta de untracked/modified para staged (pronto para commit)
    - git add * OU git add. → passa tudo do meu working directory para staging area
- git commit -m "descrição" → salvo meu código em commit (-m → flag para add descrição ao meu commit)
- git remote add origin https... → adiciono meu repositório remoto (origin é o apelido para o link do repositório remoto_convenção)
- git remote -v → lista os repertórios remotos que tenho cadastrado
- git push origin master → "empurra" para o repositório remoto citado (origin) tudo que está no branch citado (master)
- git pull origin master → "puxa" do repositório remoto para o meu repositório local
- git clone https... → clono um repositório do GitHub para o meu repositório local (com o link do repositório do GitHub) (não vem como diretório simples, vem como repositório do git)
    - git remote -v no repositório clonado → me mostra o repositório remoto ao qual está vinculado

Conflito de Merge no GitHub

Quando duas pessoas ou mais alteram o código na mesma linha. A pessoa que tentar empurrar o código para o GitHub depois da outra terá um conflito de merge, pois o arquivo que ela puxou do GitHub já não é mais a versão mais recente do mesmo no repositório remoto. Então, ela terá que puxar a versão mais recente para o repositório local (git pull) e resolver manualmente o conflito. Depois de resolvido o conflito, aí sim, ela pode empurrar para o GitHub o código atualizado.