CONHECENDO O GIT

1. Instalação do Git

Primeiro, você precisa instalar o Git em seu sistema. Você pode baixá-lo e instalá-lo no site oficial do Git: https://git-scm.com/

2. Configuração Inicial

Após a instalação, abra o terminal (no Windows, use o Git Bash que é instalado junto com o Git) e configure seu nome de usuário e email:

```
git config --global user.name "Seu Nome"
git config --global user.email "seuemail@example.com”
```

3. Criar um Repositório Git Local

Vamos criar um novo diretório e inicializá-lo como um repositório Git:

```
mkdir meu_projeto
cd meu_projeto
git init
```

4. Adicionar Arquivos ao Repositório

Adicione os arquivos ao seu repositório Git:

```
git add .
```

Isso adicionará todos os arquivos no diretório atual. Se você quiser adicionar apenas arquivos específicos, substitua o ponto por nomes de arquivo.

5. Commit de Mudanças

Agora, você precisa fazer um commit das mudanças adicionadas:

```
git commit -m "Mensagem descrevendo as alterações feitas"
```

6. Criar um Repositório no GitHub

Vá para o GitHub **(https://github.com/**) e crie um novo repositório clicando no botão "New". Dê um nome ao seu repositório e clique em "Create Repository".

7. Vincular Repositório Local ao GitHub:

Agora, vincule seu repositório local ao repositório no GitHub:

git remote add origin URL_do_seu_repositório_no_GitHub

8. Enviar as Mudanças para o GitHub

Envie suas mudanças para o GitHub:

git push -u origin master

Isso empurra seus commits para o branch master (o principal por padrão) do repositório remoto.

9. Clonar um Repositório Existente

Se você quiser trabalhar em um repositório existente do GitHub, você pode cloná-lo:

git clone URL_do_repositório_no_GitHub

10. Atualizar seu Repositório Local

Se outras pessoas fizerem mudanças no repositório remoto, você pode atualizar seu repositório local para refletir essas mudanças:

git pull origin master

11. Ramificação (Branching)

Branches são úteis para trabalhar em novas funcionalidades ou correções de bugs sem interferir no branch principal (geralmente chamado de master). Para criar um novo branch:

git branch nome_do_branch

Para mudar para o novo branch:

git checkout nome_do_branch

ou de forma abreviada:

git checkout -b nome_do_branch

Após fazer as alterações, você pode fazer o commit e mesclar (merge) o branch de volta ao branch principal.

12. Mesclagem (Merging)

Quando você completar o trabalho em um branch, pode mesclá-lo de volta ao branch principal (como master), assim:

```
git checkout master
git merge nome_do_branch
```

13. Resolução de Conflitos

Às vezes, ao mesclar branches, podem ocorrer conflitos. O Git indicará onde esses conflitos ocorreram e você precisará resolver manualmente editando os arquivos afetados. Após resolver os conflitos, você precisa adicioná-los ao stage e fazer o commit.

14. Pull Requests no GitHub

Pull Requests (PRs) são uma forma de solicitar que as alterações que você fez em um branch sejam revisadas e mescladas ao branch principal. No GitHub, vá até a página do seu repositório, clique em "Pull requests" e depois em "New pull request". Selecione o branch que você deseja mesclar e crie o pull request. Outros colaboradores podem revisar suas alterações e discutir antes de mesclar.

15. Colaboração

Você pode adicionar colaboradores ao seu repositório no GitHub para trabalharem juntos no mesmo projeto. Isso é feito nas configurações do repositório, na seção "Collaborators".

16. Gerenciamento de Versões (Tagging)

Tags são usadas para marcar pontos específicos na história do seu repositório. Isso pode ser útil para identificar versões estáveis ou marcos importantes no desenvolvimento do seu projeto. Para criar uma tag:

```
git tag nome_da_tag
```

Para enviar a tag para o repositório remoto:

```
git push origin nome_da_tag
```

17. gitignore

O arquivo .gitignore permite que você especifique arquivos ou diretórios que o Git deve ignorar. Por exemplo, arquivos de compilação, arquivos temporários, ou arquivos de configuração sensíveis.

18. Desfazer Mudanças

Se você cometer um erro e precisa desfazer as mudanças em um arquivo antes de fazer commit:

git checkout -- nome_do_arquivo

Isso descarta as mudanças não salvas no arquivo.

19. Visualizar o Histórico de Commits

Para visualizar o histórico de commits do seu projeto:

git log