Instalação e Principais Comandos do Git.

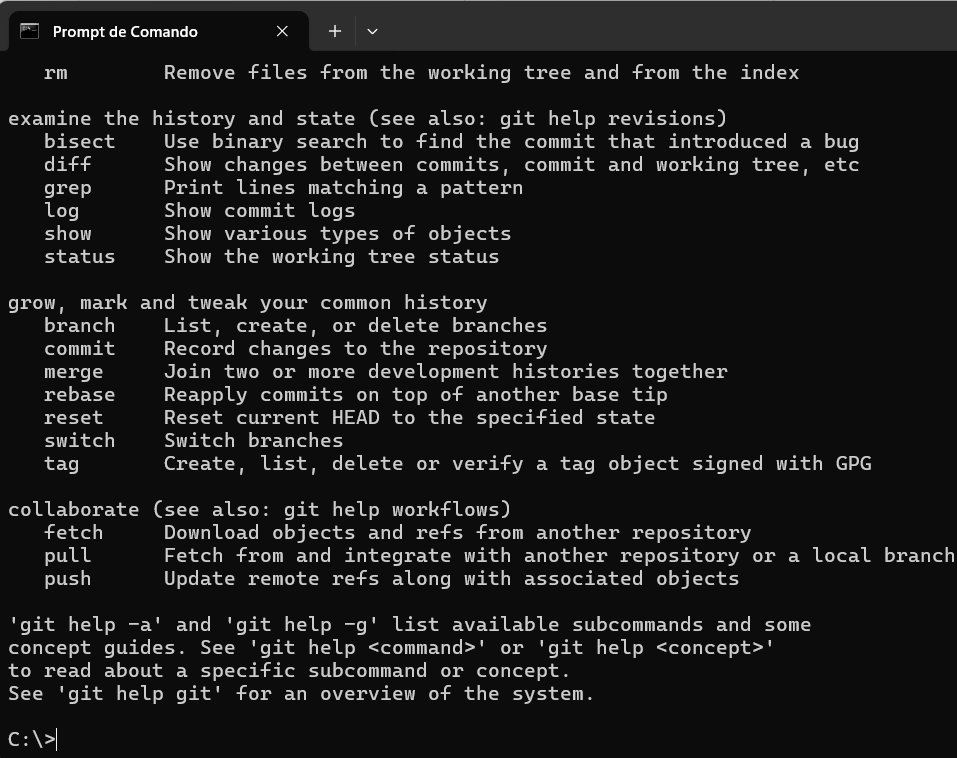
1. Baixar e instalar o Git

<https://git-scm.com/downloads>

1. Verificar se o git estar instalado.

C:\> git

Exibe nos comandos do git



1. Clonar o repositório do GitHub

Na pasta c:\>

Executar:

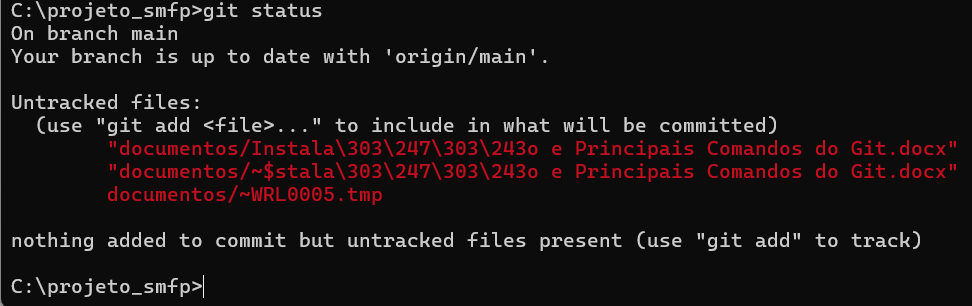
git clone https://github.com/contatojsmfp/projeto\_smfp.git

Deverá aparecer na sua maquina a pasta projeto\_smfp com todas as sub-pastas, onde você poderá fazer criação, alteração, exclusão das pastas e dos arquivos existentes nessas pastas.

1. Após o trabalho, dentro da pasta projeto\_smfp executar os seguintes comandos.

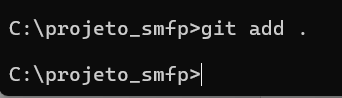
* C:\projeto\_smfp> git status

Exibe os arquivos que foram alterados, em vermelho e necessário executar o comendo “git add”



* C:\projeto\_smfp> git add .

O ponto identifica todos os arquivos alterados.



Verificando o status após “git add .”

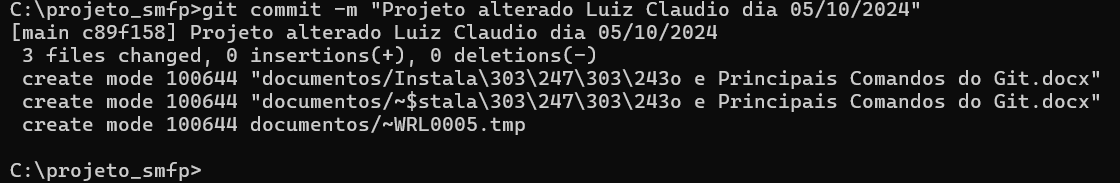
Os arquivos vão está em Verde identificando que já estão preparados para o commit.



* C:\projeto\_smfp>

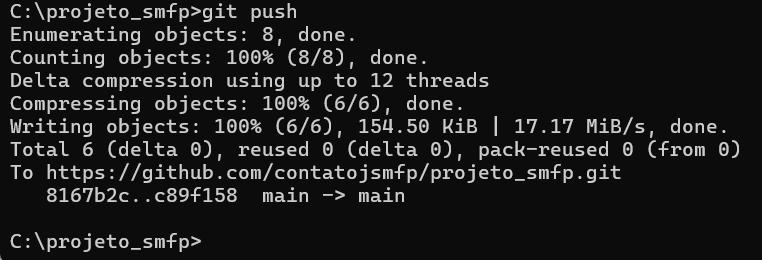
git commit -m "Projeto alterado Luiz Claudio dia 05/10/2024"

A mensagem entre aspa é um comentário da sua alteração.



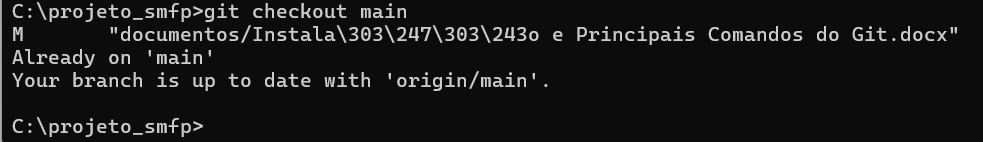
* C:\projeto\_smfp>git push

Identifica que os arquivos alterados foram enviados para o repositório.



* Antes de começar os trabalhos é necessário executar o comando “checkout” para igualar os arquivos com o repositório. “Alterações que foram feitas pelos colegas”

C:\projeto\_smfp> git checkout main



Relação dos principais comando:

Trabalho do colaborador ChartGPT.

Claro! Aqui está uma descrição dos comandos do Git que você mencionou:

1. \*\*git status\*\*:

- Este comando exibe o estado atual do repositório. Ele mostra quais arquivos foram modificados, quais estão prontos para ser commitados (staged) e quais ainda não foram rastreados pelo Git.

2. \*\*git add\*\*:

- O comando `git add` é usado para adicionar mudanças feitas em arquivos ao índice (staging area). Isso prepara os arquivos para o próximo commit. Você pode adicionar arquivos específicos ou usar `.` para adicionar todos os arquivos modificados.

3. \*\*git commit\*\*:

- O comando `git commit` salva as mudanças que você adicionou ao índice. Cada commit é como uma "foto" do estado do projeto em um determinado momento, e você deve incluir uma mensagem que descreva as alterações feitas.

4. \*\*git push\*\*:

- O comando `git push` é utilizado para enviar os commits feitos localmente para um repositório remoto. Isso atualiza o repositório remoto com as suas mudanças.

5. \*\*git checkout\*\*:

- O comando `git checkout` é utilizado para mudar entre diferentes branches ou para restaurar arquivos a um estado anterior. Com ele, você pode acessar outra branch ou desfazer alterações em arquivos.

6. \*\*git pull\*\*:

- O comando `git pull` é usado para atualizar o repositório local com as últimas mudanças do repositório remoto. Ele faz um `fetch` para baixar as mudanças e, em seguida, tenta mesclá-las automaticamente com sua branch atual.

7. \*\*git fetch\*\*:

- O comando `git fetch` baixa as mudanças do repositório remoto, mas não as mescla automaticamente. Ele atualiza sua cópia local das referências remotas, permitindo que você veja as mudanças que foram feitas no repositório remoto antes de decidir integrá-las ao seu trabalho.

8. \*\*git merge\*\*:

- O comando `git merge` é usado para unir duas branches. Quando você mescla, as mudanças de uma branch (geralmente a branch remota) são incorporadas à sua branch atual. Isso pode resultar em conflitos se houver mudanças incompatíveis que precisem ser resolvidas manualmente.

Esses comandos formam a base do fluxo de trabalho do Git e são essenciais para gerenciar o versionamento do código. Se precisar de mais detalhes ou exemplos de uso, é só avisar!