***Comandos Git***

**git config --global user.name “**nome**”** →

Comando utilizado para configurar um nome de usuário.

**git config --global user.email “**email**”** → Comando

utilizado para configurar o email do usário.

**git config --global --list** → Comando utilizado para

verificar os usuários e suas informações salvas no git.

**Obs.:** Os mesmo comandos funcionam para alterações apenas em repositório local, basta apenas tirar o **--global**.

**clear** / **CTRL + L** → Comando utilizado para limpar a tela

do git.

**pwd** → Comando utilizado para ver o diretório.

**cd ..** → Comando utilizado para voltar o diretório.

**mkdir** nome\_da\_pasta → Comando utilizado para criar

uma pasta.

**ls** / **dir** → Comando utilizado para mostrar o conteúdo do

diretório.r

**cd** nome\_da\_pasta → Comando utilizado para entrar em

uma pasta específica do diretório.

**git init .** → Comando utitlizado para iniciar um

repositório git master.

**touch** nome\_do\_arquivo → Comando utilizado para criar

arquivos no repositório git.

**start** nome\_do\_progama nome\_do\_arquivo → Comando

utilizado para criar um novo arquivo direto no progama desejado.

**git add** nome\_do\_arquivo → Comando utilizado para

confirmar ou rastrear o arquivo criado pelo **touch**.

**git add \*** → Comando utilizado para confirmar ou

rastrear todos os arquivos presentes no repositório.

**git add \*.**extensão → Comando utilizado para confirmar ou

rastrear todos os arquivos armazenados no diretório de uma determinada extensão.

**ls -l** → Comando utilizado para visualizar as

informações dos arquivos presente no diretório.

**ls -a** → Comando utilizado para visualizar os

Arquivos ocultos em um diretório.

**git status** → Comando utilizado para visualizar as

informações gerais do git criado.

**git commit -m** “ “→ Comando utilizado para gravar as

alterações no repositório, utilizando o **-m** para descrever um comentário do porque da gravação.

**git commit -a -m** “ “ → Comando utilizado para fazer um commit direto sem precisar fazer o **git add**.

**git commit --amend -m** “ “ → Comando utilizado para

fazer um commit apenas para o ultimo ponto do Stage Area.

**git log** → Comando utilizado para mostrar as informações

importantes do git.

**git log --graph --oneline --all** → Comando

utilizado para fazer um log resumido com gráfico de todos os elementos.

**git log --oneline** → Comando utilizado para ver os

commits de forma resumida.

**git log --oneline** numero → Comando utilizado para ver os

últimos commits determinados de forma resumida.

**git log --author=**nome\_do\_autor → Comando

utilizado para ver os commits de formar resumida pelo nome do autor.

**git log --before=**"YYYY-MM-DD” → Comando

utilizado para ver os commits de forma resumida por meio de uma data definida.

**git log --** nome\_da\_pasta/ → Comando utilizado para ver os

commits de forma resumida por meio de uma pasta específica.

**git shortlog** → Comando utilizado para

visualizar os commits realizados de cada autor em sua ordem de forma extramemente resumida.

**git shortlog -sn** → Comando utilizado para visualizar os autores e a quantidade de commits realizadas por estes.

**git help** / **git help -a** / **git help -g** / **git** comando →

Comando utilizado para ver as opções que o git tem de comando e obter ajuda.

**mv** nome\_do\_arquivo nome\_novo\_do\_arquivo →

Comando utilizado para renomear o nome dos arquivos no diretório.

**git rm**  → Comando utilizado para

deletar um arquivo.

**git rm --cached** nome\_do\_arquivo → Comando

utilizado para retirar um arquivo da Stage Area e volta-ló para a Área de trabalho.

**rm -rf** nome\_da pasta → Comando utilizado para apagar

uma pasta no diretório.

**git diff --staged** → Comando utilizado para ver as

informações de um ou mais arquivos editados.

**git diff** nome\_da\_branch\_ou\_hash nome\_da\_branch\_ou\_hash →

Comando utilizado para ver a diferença de duas hashs pelas posições da branch.

**git restore --staged** nome\_do\_arquivo → Comando

utilizado para retirar o rastreamento ( **add** ).

**git reset head~1** → Comando utilizado para voltar o head

para o estado anterior do **commit**.

**git rm** nome\_do\_arquivo → Comando utilizado para excluir

arquivos. **Obs.:** O arquivo vai como pedido de deleted na Staged Area.

**git log --diff-filter=D --summary** → Comando utilizado para

ver quais arquivos foram deletados e seus respectivos commits.

**git checkout** rastramento\_commit**~1** nome\_do\_arquivo →

Comando utilizado para restaurar arquivos.

**cat** nome\_do\_arquivo → Comando utilizado para ver o

conteúdo do arquivo.

**git branch** nome\_da\_branch → Comando utilizado para

criar uma nova branch.

**git branch -d** nome\_da\_branch → Comando

utilizado para excluir uma branch.

**git branch -m** nome\_da\_branch → Comando

utilizado para renomear uma branch.

**git checkout** nome\_do\_branch\_ou\_hash → Comando

Utilizado para navegar entre os branchs e hashs.

**git checkout -b** nome\_do\_branch → Comando

utilizado para criar um novo branch e navegar para ele automaticamente.

**git checkout** hash **-B** nome\_do\_branch →

Comando utilizado para fazer a alteração de um branch de um hash para outro.

**git revert** hash → Comando utilizado para desfazer as

alterações dos dados salvas no commit.

**git clone** url → Comando utilizado para clonar uma URL do

repositório remoto para o computador.

**git remote add origin** caminho\_do\_reposítório(URL)

**git branch -M main**

**git push -u origin main** → Comando utilizado para

criar um repositório no GitHub.

**git push** → Comando utilizado para enviar alterações

de um repositório local para um repositório remoto já criado.

**git push origin** branch **--force** → Comando

utilizado para forçar para a transferência dos origins de um hash para o hash atual.

**git push origin --delete** nome\_da\_origin → Comando

utilizado para deletar uma origin.

**git reset** nome\_do\_arquivo →Comando

utilizado para retirar um arquivo da Stage para a área de trabalho, sem alterar o contéudo do arquivo.

**git reset --soft** hash\_do\_commit →

Comando utilizado para voltar a um hash específico mantendo os arquivos no Stage Area.

**git reset --soft** hash\_do\_commit →

Comando utilizado para voltar a um hash específico mantendo os arquivos na Área de Trabalho.

**git reset --hard** hash\_do\_commit → Comando

utilizado para voltar a um hash específico apagando todo o conteúdo realizado no hash’s posteriores a este.

**touch .gitignore** → Comando utilizado para criar

um arquivo gitignore no projeto.

**echo** nome\_do\_arquivo **>>** **.gitignore** →

Comando utilizado para inserir um arquivo no **.gitignore**.

**Obs.:** Ao utilizar esse comando certifique-se que há dois **>>** ,pois se houver apenas 1 “**>**” o arquivo irá apenas substituir o conteúdo do arquivo.

**echo** “**\*.**tipo\_do\_arquivo” **>>** **.gitignore** → Comando

utilizado para ignorar arquivos de um terminado tipo.

**echo** “nome\_do\_arquivo**\***” **>>** **.gitignore** →

Comando utilizado para ignorar todos os arquivos que começam com aquele terminado nome.

**echo** “**?**sequencia\_especifica” **>>** **.gitignore** →

Comando utilizado para ignorar todos os arquivos que começam com um caracter e tenha uma sequência específica em seguida.

**echo**  “caminho\_do\_diretório**/**” **>>** **.gitignore** →

Comando utilizado para ignorar um diretório específico no projeto.

**echo** “**\*\*/**caminho\_do\_diretório**/**pasta\_especifica” **>> .gitignore** →

Comando utilizado para ignorar toda pasta e todos arquivos que estão dentro desta que tenha um nome específico em um projeto.

**echo** “**!**nome\_do\_arquivo” **>> .gitignore** →

Comando utilizado para não ignorar um arquivo específico depois de já ter ignorado todos os arquivos de um determinado tipo ou de um diretório específico.