1 - [COMANDOS DO DIA-A-DIA]:

Inicializar o git para rastrear arquivos locais de um projeto (sempre que iniciar um novo projeto)

```
$ git init
```

Verificar se existe alguma pendencia a ser adicionada a HEAD

```
$ git status
```

Atualiza o repositório local com as alterações publicadas/atualizadas que já estão no github
---
```
$ git pull origin main
```

Exibe o relatorio de alterações realizadas no projeto

```
$ gitk
```

Adiciona as alterações na HEAD (sempre após alterar os códigos)

```
$ git add .
```

Comitar as alterações realizada e já adicionadas (sempre após realizar o comando "git add .")

```
$ git commit -m "Mensagem aqui"
```

Publicar/subir as alterações para o github para a HEAD main (sempre após realizar o comando "git commit -m..")

```
$ git push origin main
```

---
2 - [PRINCIPAIS COMANDOS GIT]

```
$ git config --global user.name "Seu Nome"
'Configurar o nome de usuario do git'

$ git config --global user.email email@example.com
'Configurar o email do usuario do git'

$ git log
'Exibe todas as alteração realizada no projeto'

$ git log -p -1
'Exibe a ultima alteração realizada no projeto'


$ git log --pretty=oneline
'Exibe as chaves e descrição das alterações'
```

```
$ git commit --amend -m "Mensagem aqui (edicao)"
'Editar o ultimo commit'

$ git reset HEAD nomedoarquivo.extesao
'Remover um arquivo que já foi adicionado add'

$ git checkout -- nomedoarquivo.extensao
'Remover as alterações indesejadas realizadas no arquivo especificado para o último
commit, caso não tenha sido realizado o git add . ainda'

$ git checkout .
'Remover todas as alterações realizadas em todos os arquivos desde o último commit, caso
não tenha sido realizado o "git add ." ainda'

$ git rm nomedoarquivo.extensao'
'Remover o arquivo especificado localmente'

$ git rmdir
'Exclui uma pasta vazia'

$ move arquivos ou pastas de uma origem para um destino informado
git mv <origem> <destino>

$ copia um arquivo ou pasta de uma origem para um destino
git cp <origem> <destino>

$ git tag -a "1.0.0" -m "Mensagem aqui"
'Cria uma tag para marcar o projeto com a versão e uma mensagem'

$ git tag -a "0.1.0" CHAVE -m "Mensagem aqui"
'Com a chave do commit é possível criar tags de versão para commits anteriores'

$ git checkout "0.1.0"
'Exibe as informações de uma versão'

$ git tag -d "0.1.0"
'Deleta uma tag de acordo com a versão informada'

$git push origin main --follow-tags
'Subir uma tag para o github'

$ cd ../
'Voltar uma pasta no diretorio'

$ cd nomedapasta/pasta
'Navegar uma pasta dentro do diretorio'

$ git diff --staged
'Acessar o historico de alterações que já foram adicionadas, mas estão aguardando envio
para o github'

$ git reset --hard HEAD~1
'Deletar commit que ainda não tenha sido enviado para o GITHUB'

$ git reset --hard <sha1-commit-id>
'Caso queira deletar um commit específico basta usar o comando passando o id do commit'

$ git remote -v
```

'Visualizar as origens dos remotes /links do git'

$ git remote rm origin
'Remover destinos (Links) de repositorios GITHUB'

$ git clone https://github.com/... NOME-DA-PASTA
'Clonar um projeto do GITHUB para uma pasta especifica'

$ mkdir nomeDaPasta
'Criar uma nova pasta com o nome especificado'

$ touch nomeDoArquivo.extensao
'Criar um novo arquivo com o nome e extensão expecificada'

$ rm -rf .git
'Remover o arquivo .git de uma pasta que está fazendo o rastreamento das alterações'

$ git reflog
'Lista a relação de "Heads" referente as alterações comitadas, identificar o indice para o qual deseja retornar'

$ git reset HEAD@{indice}
'Retornar o projeto para uma "Head" específica, usar o comando acima para encontrar o índice desejado sem alterar os códigos'

$ git reset --hard 66c2555
'Voltar os códigos para uma versão de alteração desejada, passando o id da versão'

$ git push --force
'Realizar uma atualização forçada do código no github. Subirá mesmo se houver um conflito de versão'

Esconder um arquivo na stash (o arquivo fica congelado para o git)
$ git stash push <arquivo>

Listar arquivos que estão na stash do git
$ git stash list

Retornar o arquivo da stash (descongelar)
$ git stash pop <arquivo>

---
3 - [TRABALHANDO COM BRANCHS]

'Cria um novo ambiente e já altera o diretorio para o ambiente de teste criado'
$ git checkout -b nomedoambienteteste
Faça as alterações desejadas no projeto

Faça os commits
$ git commit -a -m "Mensagem aqui" ou "git commit -am "Mensagem aqui"

Saia do ambiente de teste e volte para o ambiente de produção
'Geralmente o nome do ambiente em produção é main'
$ git checkout main

Realize a implementação do ambiente de produção
$ git merge nomeambienteteste

deletar ambiente de teste
$ git branch -d nomeambienteteste

Criar uma nova branch no github
$ git push --set-upstream origin MinhaNovaBrach

Realizar o merge de um arquivo específico de outra branch para a main
Com a branch main ativa execute o seguinte comando
$ git checkout brachSecundaria filename.js

---
4 - [ATUALIZAR UM FORK]

'Criar uma head com o nome desejado o mais comum é usar upstream direcionando para o link originado do fork'
$ git remote add upstream https://github.com/code.......

'Realizar a atualização'
$ git fetch upstream

'Atualizar o repositório local'
$ git rebase upstream/main

---
5 - [TRABALHAR EM EQUIPE]

A maneira mais fácil de trabalhar em equipe é criar um repositório no GITHUB e convidar um usuário para ser contribuidor
1-Crie o repositório novo ou entre em um já existente
2-Vá em settings
3-Vá na aba access
4-Encontrar o usuário informando seu username e envie o convite
https://github.com/usuario/repositorio/settings/access

---
6 - [CONFIGURANDO O GIT]

Dar acesso ao computado local para acessar a conta do github
Instalar o GITBASH https://www.git-scm.com/downloads

[USAR MULTIPLOS USUÁRIOS]
https://blog.developer.atlassian.com/different-ssh-keys-multiple-bitbucket-accounts/

acessar o gitbash e digitar o comando
$ ssh-keygen

Abrir o diretorio que ele informar, dentro tera um arquivo id_rsa com uma chave
Copiar a chave e informar no perfil do github
SSH ---- > https://github.com/settings/keys > SSH and GPG keys

criar uma projeto no github com pasta e usar o comando direcionando o projeto ao link da pasta criada
$ git remote add origin https://github.com/...git

Esse comando só é executado a primeira vez, para subir o projeto para o GITHUB -> Subir o projeto inicial pelo terminal ou manual direto no github
$ git push -u origin main

Caso opte em subir o projeto manual OU o mesmo já esteja no gihub é necessário dar um gitclone em todos os terminais locais que forem trabalhar no desenvolvimento
$ git clone https://github.com/...git NOME-DA-PASTA

Atualizar o projeto local com as alterações do servidor sempre que for iniciar o desenvolvimento
$ git pull origin main

---
7 - [CRIANDO ALIAS PARA COMANDOS GIT]

Abrir o arquivo de configuração global
$ git config --global --edit

Criar uma tag de alias
```
[alias]
        s = !git status
        c = !git add --all && git commit -m
        l = !git log --pretty=format:'%C(yellow)%h%C(red)%d %C(white)%s - %C(cyan)%cn, %C(green)%cr'
```

Assim toda vez que for digitado o comando "$ git s" será executado o camando '$ git status' e assim sucessivamente

Fonte:
https://git-scm.com/book/pt-br/v1/Primeiros-passos-Configura%C3%A7%C3%A3o-Inicial-do-Git
https://www.git-scm.com/docs/git-config
https://www.youtube.com/watch?v=WVLhm1AMeYE