**Git e Github**

O Git é um sistema de controle de versões.

Existem outros sistemas de controle de versões como: CVS, SVN, Mercuviat Git é a mais comum.

Git log -> alguns dados do commit. (para sair da tela, pressione “q”).

Git log -p

Hash -> Id do commit.

.gitignore -> Arquivo o git vai ler e ignorar os arquivos com esse nome. Se quiser colocar uma pasta inteira para ser ignorada coloca: nome/

A cada alteração válida (significativa deve se gerar um commit), nuca commitar um código que não funciona.

Link com mais comandos git: <https://devhints.io/git-log>.

Git reflog -> Mostra as versões do código, como se fosse um ID de cada versão do código.

Git init --bare -> Contém somente as alterações dos commits, não uma cópia de cada versão.

Você inicializa um repositório como bare quando deseja que ele seja o repositório central.

Ele cria um caminho do repositório que vai servir como servidor do código, no meu exemplo foi: *C:/Users/N917533/OneDrive - Santander Office 365/Documentos/Git Github/servidor/git*

***(O caminho precisa estar entre aspas “ “)***

**Criando um repositório remoto :**

git remote add “**nome que vc quer dar ao seu repositório**” depois adiciona o caminho -> **C:/Users/N917533/OneDrive**

**git pull local master** – puchou os dados do diretório local da branch master

A *branch* master é a *branch* "padrão" quando você cria um repositório.

git push [repositorio] master –

git remote add origin https:// -> Adicionando um repositório para o github

git branch “nome da branch” – cria uma nova branch

git checkout “nome da branch” – muda de branch

git checkout -b "noma da nova branch" – Cria e muda a branch criada.

Branch criada, precisa fazer o seguinte

Git add . -> git commit -m “descricao do commit”

**Comandos do GIT**

Git reflog -> Mostra as versões do código, como se fosse um ID de cada versão do código.

Caso queira voltar para alguma versão: git reset --hard (id: xxxxxxxxxxxx)

**git log –graph->**  uma representação um pouco mais gráfica do *log* é exibida na linha de comando.

**Merge**

1° Checar em qual branch esta (Se tiver que trocar de branch: git checkout “nome-da-branch”).

2° git pull – Puxa as atualizações do servidor para máquina

3° git merge “nome-da-branch que vai entrar”

4° git push

**Pull Request**

Quando tem alguém que vai aprovar e mergiar nossos códigos.

**Control Z do Git**

git checkout -- “nome do arquivo” -> Aqui é como se desse um control Z no código, voltando para o estado que estava o arquivo.

git reset HEAD nome do arquivo -> Control Z de um commit(digamos que vc deu um commit add, ele volta pro estado anterior)

git reset –hard rash(id) -> Volta ou avança para uma versão do arquivo (control Z do commit).

**Salvando temporariamente**

Git stash -> Vai implementar algo no código que vai ser salvo temporariamente.

Git stash list – Vai listar todas as stash que contem para subir

Git stash drop -> apaga a ultima stash

Git stash pop -> Implementa o stash e apaga a lista (depois é necessário realizar um novo commit).

**Novo Tópico**

Git checkout rash(id) -> “Viaja no tempo”, vai para alguma versão de um commit. Quando quiser realizar alguma alteração em commit antigo, é necessário criar uma nova branch somente para essa operação.

**Para que serve o comando git checkout?** A descrição do comando git checkout --help, em uma tradução livre é: "Atualizar os arquivos na working tree para ficarem na versão especificada. [...]". Basicamente, podemos deixar o nosso código no estado do último *commit* de uma *branch*, de um *commit* específico, ou mesmo *tags* (que veremos adiante).

**Git diff xxxxxx.. yyyyyy –** Consegue ver o que foi inserido de um commit x até um commit Y

**Git diff ->** Se por acaso teve que parar no meio de onde estava no código consegue ver quais linhas faltaram incluir no commit.

**FORK**

Fork ele pega e faz uma cópia de um projeto que não é nosso e faz uma cópia pra nossa máquina

**Novo Tópico Gerando versão**

**git log -n 2 ->** Mostra somente o log das ultimas duas verões

**git tag -a “nome da versão” -m "mensagem do que quiser"**

**git tag ->** Mostra as tags que tem

**git push origin v0.1.0(nome da versão) -> Enviou como uma versão de sw, 1 release**

**Deletando branchs**

**Git branch -d “nome\_da\_brach“ ->** Comando de apagar branch, caso tenha uma branch com conteúdo que não tenha subido para master, o “d” tem que “D” maiúsculo.