\***CURSO GITHUB**

**SEÇÃO 1 – ENTENDO SOBRE O GITHUB**

**- INTRODUÇÃO**

**- HISTÓRIA DO GIT**

**- O QUE É O GITHUB** (É UM LOCAL NA WEB, NA NUVEM QUE VAI ARMAZENAR OS PROJETOS VERSIONADOS PELO GIT) O GIT NÃO O MESMO QUE O GITHUB (GIT É O SISTEMA DE CONTROLE DE VERSÃO – GITHUB É UM LOCAL ONDE VOCÊ COLOCA SEUS PROJETOS EM GIT)

**SEÇÃO 2 – CONFIGURANDO O GIT**

**- INSTALANDO O GIT**

**- CONFIGURAÇÃO INICIAL DO GIT** (GUARDA INFORMAÇÕES EM 3 LUGARES “GIT CONFIG SISTEMA” “GIT CONFIG USUÁRIO” “GIT CONFIG PROJETOS”)

\* GIT CONFIG --GLOBAL USER.NAME “PAULO MARTINS”

\* GIT CONFIG --GLOBAL USER.EMAIL “[PAULO.MARTINSCUSTODIO0305@GMAIL.COM](mailto:paulo.martinscustodio0305@gmail.com)”

\* GIT CONFIG --GLOBAL CORE.EDITOR (COMANDO DO EDITOR)

**SEÇÃO 3 – ESSENCIAL DO GIT**

**- INICIALIZANDO REPOSITÓRIO**

\* CRIAR UMA PASTA ( MKDIR GIT-COUSE )

\* ENTRAR NA PASTA ( CD GIT-COUSE/ )

\* INICIALIZAR REPOSITÓRIO ( GIT INIT)

\* SE DIGITAR (LS – LA) VAI FALAR QUE TEM UM DIRETÓRIO CHAMADO GIT

\* PARA VISUALIZAR DIRETÓRIO ( CD .GIT/)

\* VOLTAR NA PASTA (CD..)

**- USANDO EDITOR DO TERMINAL**

\* PARA EDITAR O ARQUIVO ( VI README.MD )

\* AO DIGITAR LETRA “I” PARA ENTRAR NO MODO (INSERT)

\* APÓS ESCREVER BASTA APERTAR TECLAS:

ESC/ PARA SAIR MODO

; / INICIAR ALGUM COMANDO

W/ ESCREVER E SALVAR

Q/ SAIR

**- O CICLO DE VIDA DOS STATUS DE SEUS ARQUIVOS**

UNTRACKED – ARQUIVO ADICIONADO, MAS NÃO FOI VISTO PELO GIT

UNMODIFIED – APÓS SER ADICIONADO MAS NÃO FOI MODIFICADO

MODIFIED – ARQUIVO EDITADO MAS NÃO FOI SALVO

STAGED – MOMENTO EM QUE O ARQUIVO FICA SENDO AVISADO PARA LEVAR OS ARQUIVOS A SER (UNMODIFIED) ELE ESTÁ PRONTO PARA SER SALVO.

\*SERVE PARA REPORTAR COMO ESTÁ O REPOSITÓRIO NO MOMENTO (GIT STATUS)

\* PARA CRIAR UM COMMIT (GIT COMMIT -M “MENSAGEM”) COMMIT É PEGAR OS ARQUIVOS E CRIA UMA IMAGEM

\* PARA ADICIONAR ARQUIVO ( GIT ADD “NOME DO ARQUIVO” )

**- VISUALIZANDO LOGS**

\* ( GIT LOG ) MOSTRA REST DO COMMIT, QUEM FOI O AUTOR DA MODIFICAÇÃO, A DATA E UMA MENSAGEM.

\* (GIT SHORTLOG) MOSTRA EM ORDEM ALFABÉTICA, QUANTOS COMMITS E QUAIS ELES FORAM.

\* ( GIT LOG --GRAPH) MOSTRA EM FORMA GRÁFICA O QUE ESTÁ ACONTECENDO

**- VISUALIZANDO O DIFF**

\* (GIT DIFF) MOSTRA A MODIFICAÇÃO “ USA-LO SEMPRE ANTES DE FAZER ALGUM COMMIT”

\* (GIT SHOW) MOSTRA A MUDANÇA QUE FEZ NO COMEÇO

**- DESFAZENDO COISAS**

\* (GIT CHECKOUT README.MD) ELE RETORNA O ARQUIVO PARA ANTES DA EDIÇÃO.

\*( GIT RESET HEAD README.MD) MOSTRA O ARQUIVO QUE FOI TIRADO

\*( GIT RESET - -SOFT - -MIXED - - HARD) CADA UM TEM A SUA FUNÇÃO.

**SEÇÃO 4 – REPOSITÓRIOS REMOTOS**

**- CRIANDO UM REPOSITÓRIO NO GITGUB**

\* REPOSITÓRIO REMOTO É AQUELE QUE FICA EM OUTRO SERVIDOR.

**- CRIANDO E ADICIONANDO UMA CHAVE SSH**

\* SSH = PROTOCOLO QUE AUTENTICA UM USUÁRIO REMOTO AO SERVIDOR.

\*(SSH-KEYGEN “GMAIL”) APÓS ABRIR O TERMINAL

\* (CD ~/ .SSH/) ESSE É O DIRETÓRIO ONDE FICA AS CHAVES DO SSH

\* PARA USAR A CHAVE UTILIZAR ( CAT ID\_RSA.PUB)

**- LIGANDO REPOSITÓRIO LOCAL A UM REMOTO**

\*(GIT REMOTE ADD ORIGIN [GIT@GITHUB.COM:GMAIL/GITHUB-COUSER.GIT](mailto:git@github.com:gmail/github-couser.git)) PARA ADICIONAR UM REPOSITÓRIO REMOTO.

**- ENVIANDO MUDANÇAS PARA UM REPOSITÓRIO REMOTO**

\* (GIT PUSH ORIGIN MÁSTER) IR NO TERMINAL E COLOCAR COMANDO PARA FAZER O PUSH E ENVIAR.

**- CLONANDO REPOSITÓRIOS REMOTOS**

\* (G CLONE “REPOSITÓRIO”/ GITHUB-COUSER-CLONE) COMANDO PARA CLONAR NO NOVO REPOSITÓRIO.

**- FAZENDO FORK DE UM PROJETO**

\*FORK PEGA UM PROJETO QUE NÃO É SEU E FAZ UMA CÓPIA DELE PARA VOCÊ.

**SEÇÃO 5 – RAMIFICAÇÃO (BRANCH)**

**- O QUE É UM BRANCH E POR QUE USAR ?**

\*BRANCH É UM PONTEIRO MÓVEL QUE LEVA A UM COMMIT.

\*VANTAGENS: PODE MODIFICAR SEM ALTERAR O LOCAL PRINCIPAL/ FACILMENTE “DESLIGÁVEL”/MÚLTIPLAS PESSOAS TRABALHANDO/ EVITA CONFLITOS.

**- CRIANDO UM BRANCH**

\*(GIT CHECKOUT – B TESTING) PARA CRIAR E ENTRAR

**-MOVENDO E DELETANDO BRANCHES**

\* PARA ENTRAR EM UM BRANCH (GIT BRANCH (NOME))

\* PARA VOLTAR ( GIT BRANCH MÁSTER)

\* PARA APAGAR (GIT BRANCH –D (NOME))

**- ENTENDENDO O MERGE**

\*SEMPRE QUE CRIAR UM MERGE, CRIARÁ TAMBÉM UM COMMIT

MERGE PRO – OPERAÇÃO NÃO DESTRUTIVA

MERGE CONTRA – COMMIT EXTRA/ HISTÓRICO POLUÍDO

**- ENTENDENDO O REBASE**

\* REBASE PEGA O COMMIT E MOVE PARA FRENTE (C3 PARA FRENTE DO C4)

REBASE PRO – EVITA COMMITS EXTRAS/HISTÓRICO LINEAR

REBASE CONTRA – PERDE ORDEM CRONOLÓGICA

**- MERGE E REBASE NA PRÁTICA**

\*CRIE UMA PASTA

\*INICIALIZE O REPOSITÓRIO

\*CRIE OUTRO ARQUIVO E COMMIT ELE.

\*CRIE OUTRO ARQUIVO, NO CASO O TERCEIRO.

\*USANDO O (MERGE BRANCH ‘TEST’) VAI ADICIONAR

\*(R) CRIAR UM ARQUIVO BUZZ

\* USAR SEMPRE O REBASE QUANDO PUDER

**SEÇÃO 6 – EXTRAS**

**- CRIANDO O .GITIGNORE**

\* ELE SERVE PARA IGNORAR ESPECIFICAMENTE, NÃO TRAQUEAR ALGUNS ARQUIVOS.

\* CRIAR ARQUIVO ( GIT .GITIGNORE)

\* DENTRO DO GITIGNORE, VOCÊ PODE ESPECIFICAR OQUE DESEJA

\* PARA EDITAR (VI.GITIGNORE)

**- GIT STASH É LINDO**

\*(GIT STASH APPLY) ELE VAI APLICAR AS MUDANÇAS QUE TINHA GUARDADO

\*(GIT STASH LIST) MOSTRA A LISTA DE TODOS OS STASHS QUE FOI FEITO.

\*(GIT STASH CLEAR) LIMPA TUDO QUE ESTIVER NO STASH

**- ALIAS PARA QUE TE QUERO**

\*CRIAR ATALHOS DOS COMANDOS

\*(GIT CONFIG - - GLOBAL ALIAS.S STATUS) CRIOU UM ATALHO COM O COMANDO STATUS

**- VERSIONANDO COM TAGS**

\*(GIT TAG –A 1.0.0 - -M “README FINALIZADO”) TAG CRIADA E ADICIONADA

\*(GIT PUSH ORIGIN MÁSTER - - TAGS) PARA SUBIR PRO GIT

**- SALVANDO SUA SEXTA COM GIT REVERT**

\*(GIT REVERT) ELE REVERTE O COMMIT

\* PRA QUE USAR O REVERT ? NÃO PERDER AS MUDANÇAS QUE VOCÊ FEZ.

**- APAGANDO TAGS E BRANCHES REMOTOS**

\*PARA APAGAR NO LOCAL (G TAG –D 1.0.1) MAS ISSO NÃO SOBE PARA O REPOSITÓRIO

\*PARA APAGAR AS TAGS ( GIT PUSH ORIGIN ; (A TAG QUE DESEJA APAGAR))