VERSIONAMENTO

1) INSTALAR O GIT

2) CONFIGURAR O GIT

- git config — global user.name “nomeusuario"
- git config —global user.email “colocar o email”

OBS: NO MACBOOK - configura no terminal

- git —version (para verificar se foi instalado)

OBS2: tudo tem que ser salvo na pasta do .git

OBS3: configurar o MacBook e windows para mostrar as **pastas ocultas**. No MacBook fica em “usuários - nomedousuário- .git”. No windows fica na pasta que eu criei em documentos.

OBS4: no Windows, você cria uma pasta e abre o Git Bash dentro da pasta para salvar as alterações nela.

3) NOMENCLATURAS

- Prompt
- Repositório local
- Stagging
- Commit
- Ramo principal
- Branch
- Readme
- .gitignore

4) COMANDOS

- git init : cria uma estrutura inicial do repositório do Git no computador (no windows a pasta nos meus documentos e no MacBook em “macintosh - usuários - nomedousuario”.
- git status : verifica o status da alteração no repositório.
- git add: add arquivos ao histórico do projeto, na stagging.
- git commit: registra/salva alteração no repositório.

- git log: permite visualizar as alterações feitas
- git show numero-do-commit: permite visualizar informações sobre qualquer commit.
- git remote add origin “destino”: informa a pasta remota
- git remote -v : visualiza o repositório remoto.
- git push -u origin master: publica alterações no repositório remoto
- git pull: baixa as alterações no repositório remoto.

5) PASSOS PARA ATIVIDADE NO GIT

a) iniciar o repositório (aqui vai começar a salvar naquela pasta que você escolheu) - git init
b) Criar um arquivo texto dentro do repositório. (No MacBook tem que ser no macintosh - usuários - nomedousuario - .git) -> (no Windows você cria na própria pasta que você criou.
c) Rastrear as alterações no repositório - git status
d) Adicionar versão no repositório - git add
e) Verificar alterações antes de salvar - git status
f) Salvar alterações - git commit
g) Visualizar alterações - git log

OBS1: quando eu crio uma pasta no repositório não significa que ela foi salva. Tenho que colocar ***git add*** e o nome do arquivo (na aula foi o versões.txt) para realmente salvar no stagging

OBS2: antes de salvar no commit, deve verificar o ***git status*** para ver se está tudo bem. Vai aparecer uma tela dizendo “no commit yet”mas aparece “changes to be commit”que no caso, é versões.txt.

OBS3: salvar no histórico do repositório com o ***git commit,*** que vai confirmar o documento que estava no stagging (git add) e salva no histórico de confirmação de repositórios (commit).

6) SALVAR GIT COMMIT COMANDO COMPLETO

- git commit -m “meu primeiro commit”

7) VISUALIZAR ALTERAÇÕES

- git log

- Vai aparecer o “meu primeiro commit”
- Para verificar as alterações naquele commit eu coloco “git show numero-do-commit”.

8) CRIANDO REPOSITÓRIO ONLINE

- entrar em github.com
- Criar um repositório
- Informar no terminal git remote add origin (que está no site com o nome que eu criei).
- Voce coloca **git remote add origin (link do GitHub)**
- Visualizar o repositório: **git remote -v**
- Publicar alterações: **git push -u origin master**
- Caso coloque um errado, só digitar: git remote remove origin
- No repositório vai ficar as alterações.
- Depois para salvar no local de comando: **git pull (**salva na pasta do computador).

9) CRIANDO UM .GITIGNORE

- serve para não mostrar as alterações
- Você cria uma pasta “.gitignore”mas não poder ser txt.
- Se for, coloca o código: git mv gitignore.txt .gitignore
- Ou então cria, ao invés de txt, cria com “todos os arquivos”
- Quando eu quiser que o arquivo seja ignorado, eu vou colocar dentro da pasta o nome do arquivo

9) TRUNK E BRANCH

- TRUNK é o tronco principal chamado de master.
- BRANCH é uma cópia originada do código fonte.

10) CLONANDO REPOSITÓRIO

- git pull: para baixar atualizações do repositório remoto
- git clone: clonar o repositório remoto.
  - PASSO A PASSO DO GIT CLONE
    - Acessar o git hub nosso e pegar o link (ir no repositório e depois em CODE)
    - Copiar o link e colocar: **git clone https://(link)**

11)CRIANDO UMA BRANCH

- ir no terminal e digitar:

- git checkout -b tarefa/minha-primeira-branch
- O -b é para uma branch que não existe.
- Aí você altera um arquivo que você baixou do turno
- Depois verifica o status dele com: git status
- Depois add o arquivo: git add . (Não esquecer do ponto)
- Fazer o commit: git commit -m "comentário novo"
- PUBLICAR: Empurrar a alteração para o repositório remoto: git push origin tarefa/ minha-primeira-tarefa

OBS: tudo foi feito no ramo criado agora e não no master (principal).

- **Criando outra branch depois da primeira**:
- git checkout master (voltar para a master)
- Criar uma nova branch: git checkout -b exemplo-branch
- Fazer alteração na branch

OBS: as alterações da primeira branch e nada afetam a pasta que você tem.

- git add . (Não esquecer do ponto final)
- git commit -m
- git push origin exemplo-branch

OBS: então, do primeiro arquivo que alteramos (no caso o readme) a primeira alteração era a "minha primeira branch". Nela ficou a ramificação. Depois criamos outro branch do readme sem as alterações da primeira branch com o nome "exemplo-branch".

12) UNIFICANDO BRANCH

- pode ser de duas maneiras:
    a) git pull origin tarefa/minha-primeira-branch
    b) git merge tarefa/minha-primeira-branch
- provavelmente vai dar erro quando colocar o git merge, pois temos 2 arquivos readme diferentes.
- 3 maneiras de resolver: a) incluir somente a alteração local, b) aceitar somente o que a branch alterou c) aceitar as duas alterações.
- Se aceitar as duas alterações
    - Abro o arquivo que eu alterei (no caso o readme)

    - Tiro os conflitos existentes e salvo o arquivo (tirar o conflito existente é tirar as rasuras e desenho que ficou no arquivo)
    - git add .
- git commit -m “resolvendo conflitos”
- git push origin exemplo-branch
- Ir no GitHub e ver as modificações.

13)TAGS

OBS: você tem que trocar do exemplo branch e ir para o master se quiser fazer a atividade. Para isso: git checkout master

OBS2: caso você tenha salvado a tag no lugar errado e quer apagar e fazer uma no local certo: git tag -d

- onde eu criar a tag, ela vai ficar, por isso, tem que começar demarcando o ponto atual do comando.
- git tag -a v1.0 -m “minha primeira tag”
- Para ver as tags criadas: git tag
- Para ver as informações daquela tag: git show v1.0 (o nome da tag)
- Publicar a tag para o repositório remoto: git push origin —tags