ctrl + L limpa tudo

**0. Criando um arquivo linkado ao github**

Atentem em mudar as informações de configuração abaixo para a do seu GITHUB!!!

* git init
* git add README.md
* git commit -m "Meu primeiro commit"
* git remote add origin git@github.com:seu-usuario/seu-repositorio.git
* git push -u origin master

**1. Criando pastas seguras utilizando o SSH do git, no Github.**

1. Gerar a key SSH (informe também uma senha caso queira):

$ ssh-keygen -t ed25519 -C anderson.donaruma@gmail.com

1. Mudança de diretório (cd para mudar de diretório e especificar o caminho desse diretório):

$ cd /c/Users/Ander/.ssh/

1. Listagem de arquivos do diretório (ls lista os arquivos presentes dentro da pasta que você está):

ls

1. Verificação do SSH (cat é um comando para visualizar o conteúdo dessa chave SSH. Dentro do git hub vá em profile/settings/SSH and GPG Keys -> crie um arquivo SSH):

$ cat id\_ed25519.pub

1. Conferir o caminho que você está (pwd serve para mostrar o caminho completo da pasta que você está):

$ pwd

1. Abrir o gerenciador de chaves SSH (inicializar o agente responsável por lidar com as chaves SSH):

$ eval $(ssh-agent -s)

1. Informar a chave SSH que foi criada (vamos passar para o agente a chave que ele deve utilizar. Coloque a senha que foi informada na criação da chave SSH passo 1):

$ ssh-add id\_ed25519

FEITO

Agora que você criou uma conexão SSH segura com o servidor do github, você pode clonar projetos para dentro dessa pasta ssh

**2. Comandos com o git**

Alguns comandos que utilizaremos serão:

* git init
* git add
* git commit

Criar diretório -> $ mkdir nome-do-diretório

1. dentro do diretório utilizar o git init. Ele criará uma pasta oculta chamada “.git”

$ git init

1. A pasta .git que foi criada pode ser vista no terminal do git com o comando ls -a

$ ls -a

1. cd .git/ (para entrar na pasta “.git”
2. $ ls (listando os itens dentro do diretório “.git”)

HEAD config description hooks/ info/ objects/ refs/

1. Para retornar um nível utilizar dois pontos no comando cd

$ cd ..

1. Se é a primeira vez que você está usando o git ele vai pedir algumas configurações iniciais:

$ git config --global user.email "anderson.donaruma@gmail.com"

$ git config --global user.name andy-spch

1. Para utilizar níveis de título chamados de Markdown utilizamos até o nível 6 sendo:

# nível 1 – equivale ao <h1></h1> do HTML TEXTO BEM GRANDE

## nível 2 – equivale ao <h2></h2> do HTML

### nível 3 – equivale ao <h3></h3> do HTML

#### nível 4 – equivale ao <h4></h4> do HTML

##### nível 5 – equivale ao <h5></h5> do HTML

###### nível 6 – equivale ao <h6></h6> do HTML texto menor

1. Coloquei uma receita de macarrão com camarão no arquivo txt
2. Agora vamos colocar os arquivos na nuvem do github:

$ git add \*

$ git commit -m "commit inicial de receitas"

**Cria nova Branch**

Git checkout -b nome-da-branch (primeira maneira de fazer a criação de uma branch)

Git branch nome-da-branch (segunda maneira de fazer a criação de uma branch)

**Deleta uma Branch**

git branch -d nome-da-branch

**Renomeando pastas ou arquivos**

mv ‘nome da pasta ou arquivo’ ‘novo nome da pasta ou arquivo’

**Atualizar arquivos do pc utilizando o git pull**

git pull origin master – para a branch master

git pull origin ‘nome da branch’ – para uma branch secundária

**Clonando repositórios**

git clone ‘link do repositório’

**Verificando os logs de commits**

git log – log de commits completo

git log --oneline – log de commits abreviado

**Configurar e-mail e username do git na máquina**

git config --global user.email

git config --global user.name

git config --list – para ver a lista das configurações gerais da sessão

git config --global --list – para ver a lista das configurações de e-mail e name da sessão

**Liberar acesso de pessoas para realizar alterações diretamente no seu repositório**

Você pode liberar o acesso de pessoas que estejam trabalhando junto com você no mesmo projeto para que façam as alterações diretamente no seu perfil. Para isso a pessoa enviar uma solicitação de acesso ao seu repositório no GitHub, e você como dono do repositório terá a liberdade de escolher se essa pessoa terá ou não acesso e permissões para realizar alterações no repositório.

**Pull Requests**

Você pode clonar um repositório aberto ao público.

A convenção pede para que você clone o repositório, e após isso faça uma branch para informar a alteração que será feita.

Feito isso você pode fazer o processo padrão (git add . > git commit -m “descrição” > git push origin “nome da branch”) para o seu repositório no GitHub e lá **dentro do GitHub você solicita o Pull Request**.

Essa Pull Request aparecerá para o dono do repositório e ele poderá avaliar e solicitar alterações, antes de incorporar (merge) no repositório principal dele.

**Configurando Aliases**

**Criando aliases:** git config --global alias.”abreviatura” “qual o comando”

**Exemplo:**

git config --global alias.s status (estou substituindo “git status” para “git s”)

**Verificando os Aliases já criados**

Digite git config --global –list, para ver os Aliases já criado.

**Removendo aliases:** git config –global –unset alias.”abreviatura”

**Exemplo:**

git config –global –unset alias.s (estou removendo o alias “s” do “git status”)

**Saindo da tela secundária chata**

Aperte **ESC** e depois execute o comando **:wq**

**Resolvendo o problema do pull em históricos diferentes de commits, branches, etc.**

O erro fatal: refusing to merge unrelated histories geralmente acontece quando você tenta fazer o git pull de um repositório remoto, mas o seu repositório local possuí um histórico de commits, branches, etc, diferente do que está no repositório remoto.

Para permitir que o Git faça o merge de dois projetos com históricos diferentes, é só passar o parâmetro --allow-unrelated-histories quando for fazer o pull, assim:

git pull origin master --allow-unrelated-histories

(lembre-se de trocar os nomes do remote e branch para os que você utiliza)

Comando para forçar históricos diferentes:

git pull origin master --allow-unrelated-histories

Criando um repositório no Github

Vá até a pagina de repositórios do Github e navegue até o botão de criar repositório. Complete o processo de nomeio do repositório e arquivos e clique em criar repositório.

Pegue o arquivo https do local do arquivo no Github.

Você vai entrar no seu Shell de preferência e vai digitar os seguintes comandos:

- **git remote add origin “link https”** (utilize **git remote -v** , para saber o status)

- **git push -u origin master**