Github -> local para armazenamento de projetos versionados pelo git.

Git-> sistema para controle de versão

-------------------------------------------------------------------------------------------------------------------

INSTALANDO O GIT

\* No linux já vem instalado

-------------------------------------------------------------------------------------------------------------------

CONFIGURAÇÃO INICIAL DO GIT

O fit guarda as informações em 3 lugares (gitconfig do sistema, do usuario e do projeto)

\* Config do usuário conhecido como **global**

**$git config --global user.name “Aline Lopes”**

**$git config --global user.email “aline.marina94@gmail.com”**

*Não mostra nada, configura ocultamente*

CONFIGURAR EDITOR PRINCIPAL DO GIT

**$git config --global core.editor code .**

*DESCOBRIR INFORMAÇÕES DA VARIAVEL*

***$git config user.email*** *ou* ***user.name***

Saber todas as informações do git

**$git config --list**

-------------------------------------------------------------------------------------------------------------------

INICIALIZAR O PROJETO

Criar pastas

**$mkdir git-course**

Entrar na pasta

**$cd git-course**

Responsavel por inicializar o repositório e enxergar as modifcações do projeto

**$git init**

Entrar no .git criado

**$cd .git**

Pastas do .git

HEAD=branche padrão que ele está

Branches=quais os branches

Config= configuração do repositorio

Description=descrição

Hooks=gatilhos para certas ações

Info=

Objects=

Refs=

-------------------------------------------------------------------------------------------------------------------

O CICLO DE VIDA DOS STATUS DE SEUS ARQUIVOS

Separa em quatro estados bem definidos como os arquivos serão divididos

UNTRACKED= o arquivo acabou de ser adicionado no repositório, mas o git ainda não sabe que ele existe

UMMODIFIED= não foi modificado. Existe mas não houve modificaão.

MODIFIED= arquivo modificado

STAGED= momento em que o arquivo é armazenado para as versões.

Uma vez que ele foiadicionado no staged, o status altera para UMMODIFIED

**$git status**

**$git add git\_e\_github.docx**

Se alterar o arquivo antes do commit e quiser que as modificações sejam comitadas e necessário add novamente

**$git commit -m "Add tutorial até commit"**

Os números entre as chaves siginificam o id da versão.

Se comitar sem add dá erro, ele não é comitado.

-------------------------------------------------------------------------------------------------------------------

VISUALIZANDO LOGS

Mostrar informações sobre o commit

**$git log**

Informações sobre de que branche p que branche, quais tags foram adicionadas

**$git log --decorate**

Filtrar pelo autor

**$git log --author="Aline"**

Em ordem alfabetica mostra quem foram os autores, quantos commites fizeram e quais eles foram

**$git shortlog**

Quantidade de commites e a pessoa

**$git shortlog -sn**

Mostra em forma gráfica o que está acontecendo com o branches e versões

**$git log --graph**

Pela hash dá p saber o que foi modificado, o que foi commitado ou não

**$git show *0d7a9b8bc2493759d187c64335b996dfe31972d6***

*aline@aline-PC:~/git-course$ git log --graph*

*\* commit 0d7a9b8bc2493759d187c64335b996dfe31972d6 (HEAD -> master)*

*Author: “Aline <“aline.marina94@gmail.com”>*

*Date: Thu Aug 2 18:03:50 2018 -0300*

-------------------------------------------------------------------------------------------------------------------

VISUALIZANDO O DIFF

Visualizar as alterações do arquivo antes de dar commit.

**$git diff**

Saber qual os arquivos que foram modificados

**$git diff --name-only**

Quando se da commit em um arquivo que já existiu pode-se utilizar o *-am*(adicionar todos os arquivos modificados mais a mensagem) Não é preciso utilizar o *add.*

**$git commit -am "Edit arquivo"**

-------------------------------------------------------------------------------------------------------------------

DESFAZENDO COISAS

Resetar arquivo antes de add

**$git checkout git\_e\_github.docx**

Resetar arquivo depois do add. Tirar o arquivo do unstaged.

**$git reset HEAD git\_e\_github.docx**

*Unstaged changes after reset:*

*M git\_e\_github.docx*

Existem três tipos de reset que são:

--soft = pegar as moficações e ignorar o commit e vai deixar ele com o status unstaged, pronto para ser commitado novamente

--mixed = mata o commit e voltar os arquivo para o status mofied

--hard = ignorar a existência do commit e tudo que foi ,feito do commit.

Para fazer o reset, é necessário escolher sempre a hash do commit antes do deseja-se ignorar.

Exemplo:

Commit 3

Commit 2

Commit 1

Se quer ignorar o Commit 3, escolhe a hash do Commit 2.

**$git reset --soft 64573gt59** (64573gt59 representa o hash)

Obs: --hard não deve ser usado em repositórios compartilhados.

-------------------------------------------------------------------------------------------------------------------

CRIANDO O REPOSITÓRIO REMOTO

Ir no site do github e criar um novo repositório apertando o botão

*Não selecionar a opção para gerar um readme.mp pois ao se ter histórico local, o historico gerado por essa info será diferente. Mlehor evitar tal situação.*

-------------------------------------------------------------------------------------------------------------------

CRIANDO E ADICIONANDO UMA CHAVE SSH

Obs: Para problemas de push para o repositório remoto.

O SSH serve para autenticar o usuário no servidor. A chave privada presente na nossa máquina é utilizada para descriptografar a chave pública do servidor e assim permitir submeter o arquivo ao repósitorio remoto.

*GERAR A CHAVE SSH*

*Abrir o terminal e colocar*

ssh-keygen -t rsa -b 4096 -C "*your\_email@example.com*"

***Obs:*** *o email precisa ser exatamente igual ao do usuário*

Aparece o texto

Generating public/private rsa key pair.

Local para adicionar a chave. Geralmente deixa como padrão.

Enter a file in which to save the key (/home/*you*/.ssh/id\_rsa): *[Press enter]*

Quando pede um passaphrase e clica em enter. Não se preocupe com isso.

Enter passphrase (empty for no passphrase): *[Type a passphrase]*Enter same passphrase again: *[Type passphrase again]*

É possível encontrar as chaves SSH no diretório: cd ~/ .ssh/

Liste as chaves criadas com ls. A chave que iremos precisa possui o final .pub. Para utilizá-las é necessário:

**$cat id\_rsa.pub**

Assim terá acesso a chave. Copie a chave. Vá ao github.

Selecione no menu *settings.* Selecione SSH and GPG Keys. Aperte o botão SSH Key. Adicione a chave copiada e adicione um título.

-------------------------------------------------------------------------------------------------------------------

LIGANDO UM REPOSITÓRIO LOCAL A UM REMOTO

Informações para a ligação são dadas na pagina do repositório criado. Siga-as:

**$git remote add origin https://github.com/AlineLopes/git\_course.git**

**$git remote**

**$git remote -v**

O git push envia todas as informações para o repositório criado remotamente

**$git push -u origin master**

Ele diz que vai enviar os dados do respoitorio local(master) para o repositorio remoto(origin)

-------------------------------------------------------------------------------------------------------------------

ENVIANDO MUDANÇAS PARA UM REPOSITÓRIO REMOTO

**$git status**

**$git commit -am "Add instrucoes"**

**$git push origin master**