Um repositório armazena arquivos como uma estante em pequenos pacotes, no qual chamamos de **commit**.

Cada um desses pacotes (**commit’s**) tem uma data de criação e um autor.

Os commits serão o nosso histórico de mudanças que foram feitas no projeto.

Para se **criar** um **repositório local**, baste dar o comando: **git init**

Local => está no seu computador

Remoto => se hospeda no GitHub

Para adicionar sua identidade a um repositório você precisar dar um comando git:

**git config --global user.name "seunome"**

* Configura o nome que você quer ligado ao commit

**git config --global user.email "seuemail"**

* Configura o e-mail que você quer ligado ao commit

Todo repositório criado no seu computador vai ter seu nome e seu e-mail associado.

Para apontar o repositório remoto (GitHub), é necessário utilizar o seguinte comando:

**git remote add origin https://endereço\_do\_repositório**

O comando **git**, amostra todos os comandos do git.

O **git log** é o comando que traz o rastreio de tudo que foi mudado.

Adicionando arquivos ao repositório

**git add** // adiciona os arquivos

**git status** // mostra o estado dos arquivos

Configuração para múltiplos arquivos

**git add . // adiciona todos os arquivos ao repositório**

**git status // lista todos os arquivos e alterações**

**git commit –m “adicionando todos os arquivos” // Adiciona comentários ao commit**

O commit adiciona data + mensagem + autor na alteração.

Salvando os arquivos

Os commits são pontos cronólogicos na linha do tempo do projeto que nos permitem ter uma visão da evolução daqueles arquivos ao longo do tempo.

Para fazer um commit devesse primeiro adicionar os arquivos ao repositório, e depois executar o commit.

**git add .**

**git commit –m “mensagem”**

Aqui você cria um ponto na linha do tempo do seu projeto.

Aula 01 – Git [encontro]

O que seria um Sistema de Controle de Versão?

R: São ferramentas de software que ajudam as equipes de software a gerenciar alterações ao código-fonte de um projeto ao longo do tempo.

O que é um commit?

R: O commit significa pegar todos os aqrquivos que estão naquele local, que foi adicionado pelo comando add, e criar uma revisão com um número e com um comentário, que será visto por todos. Para fazer um commit devesse primeiro adicionar os arquivos ao repositório, e depois executar o commit.

Qual a diferença entre Git (local) e Github(remoto)?

R: O Git é um sistema de **controle de versão de arquivos**, muito utilizado no desenvolvimento de software onde diversas pessoas estão contribuindo simultaneamente, podendo criar e editar arquivos. Já o GitHub é um plataforma onde você pode **armazenar** seus projetos. Geralmente ambos **trabalham juntos**, onde o Git faz o **controle da versão dos arquivos**, e quando está tudo certo, é **enviado** para o GitHub, para **armazená-los.**

Revisando

O que é preciso para usar o Git?

R: Para utilizaro Git é necessário baixar o Git Bash

Qual o comando para verificar modificações?

git status

Quais são os passos para realizar um commit?

1 – git init

2 - git config --global user.name "seunome" / git config --global user.email "seuemail". Com isso, se adiciona sua **identidado** ao **repositório**.

3 – git add . / 4 – git status / 5 – git commit –m “comentário do commit”.

Com o commit ‘’fecha’’ o pacote e enviar.

<https://jamboard.google.com/d/1jPnje0tqwDvzBGGhT7meCJETjiCYA94eEMFp06Mn-6Y/edit?usp=sharing>

Link do exercício proposto na aula de 16/08 (Segunda-feira).

GitHub (Aula 5)

Repositório remoto

Repositório remoto é um lugar onde guardamos os arquivos de um projeto. 1 projeto corresponde a 1 repositório no GiHub, e podesse ter quantos repositórios quiser.

Os repositório que estão no **GitHub** chamados de **repositório remoto,** já que estão na nuvem. Em paralelo, cada membro da equipe faz uma cópia desse repositório no seu comutador, o qual chamado de **repositório local** **(Git).**

**>\_git remote add origin urldorepositórioremoto**

Instância o endereço do repositório remoto (GitHub)

**>\_ git config –list**

Serve para visualizar a **configuração** do Git

**>\_ git push**

Esse comando solicita que os arquivos que estão **‘’comitados’’** em sua máquina sejam **‘’empurrados’’**, ou seja, carregados em seus repositório.

Porém ele precisa de um complemente que é:

**>\_git push origin main**

Em outubro de 2020, o Github mudou o nome do branch master para os novos repositórios. Agora, este ramo é chamado de **principal (main)**. Portanto, seu comando deve ser como mostrado acima.

Após digitar estes comandos, o terminal irá solicitar:

**Seu username e sua senha**

Se não retornar nenhuma mensagem de erro, basta verificar em sua página do repositório os seus arquivos.

Para baixar a última versão dos arquivos de um repositório, utilizamos o comando **>\_ git clone** [**https://github.com/dougsn/meurepositorio**](https://github.com/dougsn/meurepositorio)

Com isso, o clonado os arquivos do diretório especificado no repositório local.

Para atualizar os arquivos de um repositório que já se encontra em sua máquina e foram modificados por outros colaboradores, utilizar o comando: **>\_ git pull origin main urldorepositórioremoto**

Sendo assim, os comandos principais para baixar os arquivos de um repositório:

**Clone**: cria uma cópia local de um repositório remoto.

**Pull**: atualiza os arquivos com as modificações feitas por seus colegas.

Quando se trabalha com um projeto que já tem um repositório remoto, não é necessário dar um **git init,** pois com **o git clone**, ele já vem ‘**’embutido’’**, pois quem criou o **repositório**, já deu o **comando git init**.

O arquivo **README**, é um arquivo (txt) de descrição longa do projeto.

Git + GitHub (Aula 06)

Criação do repositorio da mochila do viajante no github.

Clonagem do repositório no diretório Desktop com o **git clone**.

Abertura do diretório no GitBash e troca de branch com o: **git branch “nomeDaRamificação”**, para sair do (main) e ir para uma ramificação que esteja separada do branch principal que é o main.

Após a criação do **brach(Ramificação)**, fui utilizado o **git checkout “nomeDaRamificaçãoCriada”**, para sair da ‘’linha do tempo’’ principal e ir para a ramificação criada pelo **git branch**.

Com isso, já nos encontramos na ‘**’linha do tempo’’ do branch criado**, assim, podendo fazer as modificações e testes que quiser.

Depois, para a adicionar o que foi criado na ‘’linha do tempo’’ principal (main), se utiliza o **git add .**

Após isso, se prepara o **commit -m “comentário do que foi feito”**. Depois é necessário **‘’avisar’’** o GitHub que foi criado uma **branch(ramificão),** com o comando **git push --set-upstream origin “nomeDaBranch”**. Após isso, é necessário **confirmar o merge (mescla) entre as “2 linhas do tempo”**, a **principal (main) e a criada (Douglas\_Nascimento)** no **GitHub.** Após unir, a branch será adicionada ao seu GitHub e os arquivos que foram criados e modificados, foram incluido no seu repositório, tanto no **principal (main) e branch criada (Dougla\_Nascimento).**

O comando **git push --set-upstream origin “nomeDaBranch”** serve para enviar a branch para o repositório remoto (GitHub).