*Let's q bora!!*

**O que é git?**

É uma ferramenta de versionamento, usada principalmente para código, embora seja possível armazenar versões de qualquer tipo de arquivo. Tem como objetivo armazenar as versões de um código/software e as informações sobre elas, por exemplo, quando ocorre uma mudança, queremos saber quem a fez, porque a fez, quando a fez, etc. O git possibilita a navegação entre as versões de um código e, além disso, torna possível a criação de mais de uma linha de desenvolvimento.

**GitHub**

Enquanto o Git é a ferramenta que versiona os códigos/arquivos, o GitHub é um servidor de repositórios Git, ou seja, é possível hospedar seu repositório Git lá e compartilhá-lo com mais pessoas. O GitHub atualmente é bem importante na vida do programador, já que muitas empresas o utilizam para conhecer o portfólio dos candidatos à uma vaga.

Além disso, também é tido como uma rede social de programação, é possível acessar códigos de grandes empresas, como Facebook, Google, etc., além de projetos Open Source.

**Instalação do git**

https://git-scm.com/book/pt-br/v2/Come%C3%A7ando-Instalando-o-Git

**Chave SSH e documentação git**

Vamos ver como conectar uma máquina ao servidor que guarda o repositório remoto, nesse caso o GitHub. Há duas formas de se fazer isso, a primeira é através de https, o que ocasionaria em precisar colocar usuário e senha sempre que se quisesse registrar alguma mudança. A outra forma, que nós vamos usar, é a chave SSH, que permite enviar os dados ao repositório remoto sem necessidade de autenticação.

Links para gerar chave SSH e conectar no github:

Gerar chave:<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

Add chave no github:<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account>

Testar conexão SSH:<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/testing-your-ssh-connection>

**Comandos do git:**

**git help**: printa no terminal informações sobre os comandos

git help <command>

**git init**: criar um repositório git

git init <nomeRepositorio>

**git add:** adiciona um arquivo ao git, ao versionamento

git add <nomeArquivo>

git add .

**git commit**: registra as mudanças feitas no diretório

git commit -m “mensagem de commit”

Na primeira vez que se tentar fazer o commit, o git vai perguntar quem é você, precisa usar dois comandos para colocar usuário e e-mail do usuário:

git config –global user.name “userName”

git config –global user.email “email@emaill.com”

Recomendável colocar os mesmos dados do GitHub.

**git status:** para ver o status do repositório

g**it log**: para ver os commits que foram feitos

**git diff:** exibe diferenças entre commits e branchs

git diif HEAD~1

OBS: as mudanças ficam mais visíveis e interpretáveis quando observadas pelo GitHub.

**Conectando ao GitHub:**

**git remote:** conseguimos conectar o repositório a um repositório remoto

git remote add origin git@github.com:usuarioGitHub/nomeRepositorio.git

git remote -v: lista os remotes que temos, vai aparecer duas linhas por remote no terminal (fetch e push), o fetch baixa as coisas do repositório remoto e o push envia as coisas para o repositório remoto.

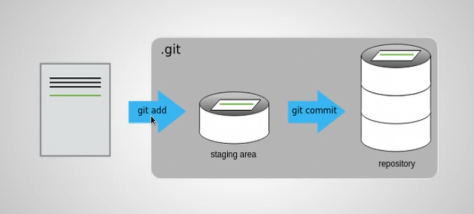
**git push:** envia os commits para o repositório remoto

git push -u origin master na primeira vez, para setar a linha master para fazer o envio das informações ao repositório remoto, depois continua com git push

quando se cria branchs novas, vai precisar de um comando para criar essa branch no repositório remoto também.

**Estados dos arquivos:**

* não monitorado (untracked): quando ainda não está no git
* modificado (modified): quando já está no git e é modificado, mas sem adicionar ao git a modificação, ou seja, antes do git add
* preparado (staged): depois do git add, já foi colocado no git, mas sem comitar
* consolidado (commit): quando o arquivo já foi commitado, quando a modificação já foi “declarada”



**Workflow do git:**

1. **git init**, cria repositório

2. cria/edita o arquivo

3. **git add <**nomeArquivo>**,** adiciona o arquivo ao git

4.**git commit**, salva as mudanças feitas no git

Na primeira vez que se tentar fazer o commit, o git vai perguntar quem é voê, precisa usar dois comandos para colocar usuário e e-mail do usuário:

git config –global user.name “userName”

git config –global user.email “email@emaill.com”

É recomendável colocar os mesmos dados do GitHub.

OBS: sem o –global, os dados são definidos apenas par o repositório atual, com o global ele define esses dados para todos os repositórios do computador.

**Conexão com GitHub**

5. **git remote**

git remote add origin git@github.com:usuarioGitHub/nomeRepositorio.git

6. **git push,** envia os commits (modificações) para o repositório remoto

git push -u origin master na primeira vez, para setar a linha master para fazer o envio das informações ao repositório remoto, depois continua com git push

quando se cria branchs novas, vai precisar de um comando para criar essa branch no repositório remoto também.

7.**README.md** é utilizado pelo GitHub como uma ferramenta de estilização para, por exemplo, mostrar coisas na página inicial do repositório, .md é uma extensão de arquivo chamada markdown.

**Documentação git:** <https://git-scm.com/>

**Atividades e prática:**

* Primeira ideia:

Inicialmente, fazer algo simples para utilizar os comandos mesmo, como criar repositórios, arquivos nesses repositórios, editar os arquivos, fazer a conexão e enviar os commits para o GitHub.

* Segunda ideia:

Posterior a isso acredito que seria legal uma atividade já cooperativa, pensei em uma, por exemplo, que daria para fazer com documentos .txt, no qual cada documento pode ter um número de palavras limite e os documentos contam uma história, cada pessoa escreve uma partezinha da história - como naquelas rodas de acampamento onde cada um emenda um pedaço de história - e vai pro próximo, sempre que chega no limite de palavras é necessário criar um novo .txt. Cada .txt poderia ser um capítulo ou algo do gênero, até formar pelo menos uns 3 ou 5 capítulos. Acho que como ponta pé inicial para uma ideia de trabalho em equipe poderia ser bom, para ter um primeiro contato em usar o git e GitHub de forma cooperativa.

* Terceira ideia:

Também acho que seria legal abrir um espaço depois para colocarmos a mão na massa de forma mais autônoma, no sentido de todo mundo dar ideias de como isso poderia ser usado e tentar fazer com que fosse. Mesmo que com partes de coisas não vistas nos conceitos ali de cima, por exemplo, com coisas de git ainda não usadas, as quais teríamos que pesquisar e procurar sobre para dar um jeito de concluir a ideia.

OBS: não pensei que necessariamente as 3 atividades precisassem ser direto no primeiro encontro, principalmente a segunda. Mas no caso da terceira, acredito que poderia ser algo comum dos encontros, com o que tivermos estudando no momento.