**O que é git?**

É uma ferramenta de versionamento, usado 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.

**Instalção git (linux)**

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”

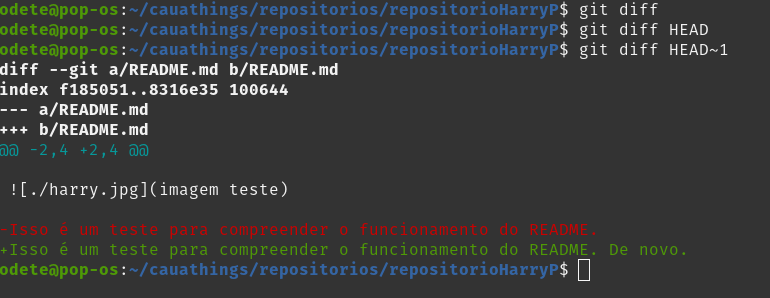
git config –global user.email “[email@emaill.com](mailto:email@emaill.com)”

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

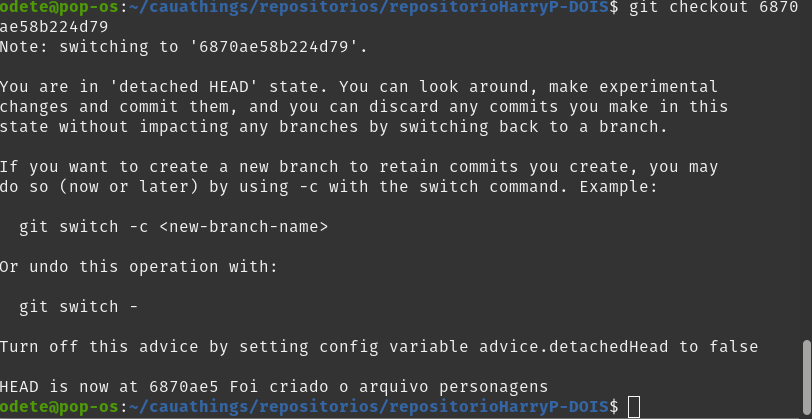
 Nesse caso não acontece nada só com git diff e git diff HEAD porque o ambos esses comandos se referem somente ao commit atual, com o HEAD~1, o comando compara o commit atual com um commit antes.

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

g**it clone <URL>:** baixa um repositório remoto, já vem com o remote configurado.

g**it pull:** baixa as alterações feitas no repositório remoto.

g**it checkout <commit> <file>:** permite ver como um arquivo ou todo o repositório estava em determinado commit.

Quando se faz isso, aparece essa mensagem, que diz que agora, com o checkout, você saiu da ponta dos commits atuais em que estava e voltou a algum anterior, neste modo não é possível fazer commits, a menos que se faça uma nova branch a partir dali. **Git checkout master:** volta para a branch master, como estava anteriormente, sendo assim possível voltar a commitar.

**Desfazendo alterações:**

* Com checkout
  + git checkout -- <path or file>: desfaz as mudanças feitas que ainda não estão no stage, ou seja, as mudanças que foram feitas, mas não foram adicionadas ao git com o git add.
  + Git checkout HEAD -- <path or file>: desfaz as mudanças incluindo o stage (pós git add).
* Com revert
  + git revert <commit>: cria um novo commit que desfaz as alterações do commit especificado.
* Com reset, não é interessante usar após o git push, posi alguém já pode ter baixado as modificações feitas pelo commit que se deseja apagar/reverter
  + git reset <commit>: reseta o repositório para determinado commit, como se apagasse o commit desejado.

**Conectando ao GitHub:**

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

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

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

**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.

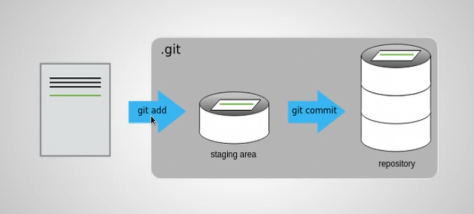
**Estados dos arquivos:**

-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/clone**, cria repositório ou baixa um repositório remoto

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 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](mailto:email@emaill.com)”

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.

**Conflitos**

Acontece quando, em projetos cooperativos. Ocorre quando são unidas duas versões de código com as mesmas linhas editadas de diferentes formas. O git identifica os conflitos e aguarda a solução deles.

**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/

**Exercícios:**

1. criar um arquivo

2. editar o arquivo

3. criar outro arquivo

4. editar o outro arquivo

5. criar outro arquivo sem função nenhuma e apagá-lo depois

A cada mudança feita no repositório (criar, editar e apagar arquivos), usar o git status e, após verirficar se a mudança está correta, fazer o commit dela. Usar o git log após cada commit.