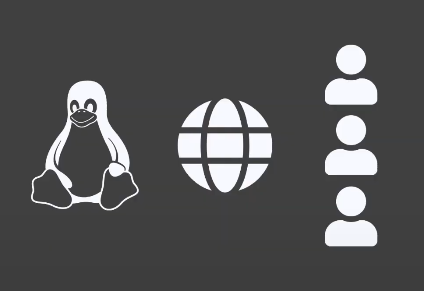
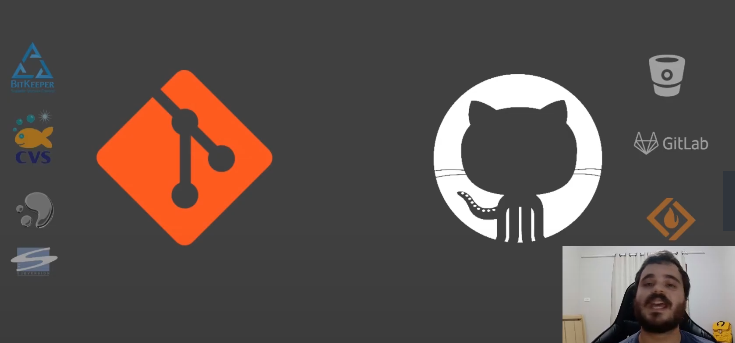
O Git foi criado em 2005 pelo “Linus Torvalds” criador do Lunux. O Linux foi feito de forma colaborativa, equipes distribuídas no mundo inteiro.

O Git é um software de versionamento de códigos.



 Git e GitHub são duas tecnologias distintas “diferentes”, uma complementa a outra mais não a mesma coisa.

 Existes vários outros softwares de mesmo propósito.

**OS BENEFÍCIOS DE ESTÁ APRENDENDO ESTAS DUAS TECNOLOGIAS JUNTAS:**

1. Controle de Versão
2. Armazenamento em nuvem
3. Trabalho em equipe
4. Melhorar seu código
5. Reconhecimento

**Navegação via command line interface e instalação**

|  |  |  |
| --- | --- | --- |
| **Comandos Windows** | **Comando Linux** | **Para que serve** |
| **dir** | **ls** | **Listar arquivos e diretórios** |
| **cd** | **cd** | **Acessar um diretório** |
| **cd ..** | **Cd ..** | **Volta para a pasta anterior** |
| **cls** | **clear [atalho(ctrl+L)** | **Limpa o terminal** |
| **tab** | **tab** | **Auto completar** |
| **mkdir** | **mkdir** | **Cria uma pasta** |
| **echo hello > hello.txt** | **echo hello > hello.txt** | **Cria um arquivo de texto** |
| **del workspace** |  | **Deleta somente os arquivos contidos dentro da pasta workspace.** |
| **rmdir /S /Q** | **rm -rf workspace/** | **Deleta a pasta woskspace e tudo que estiver dentro dela.** |

**Instalando o GIT no Linux Ubuntu**

1. sudo apt-get install git

**Entendendo como o Git funciona por baixo dos panos**

**- SHA1**

**- Objetos fundamentais**

**- Sistema distribuído**

**- Segurança**

**A sigla SHA significa Secure Hash Algorithm (Algoritmo de Hash Seguro), é um conjunto de funções hash criptográficas projetadas pela NSA (Agência de Segurança Nacional dos EUA).**

**A encriptação gera conjunto de caracteres identificador de 40 dígitos.**

<https://pt.wikipedia.org/wiki/SHA-1>

1. SHA1("The quick brown fox jumps over the lazy cog")
2. resulta no hexadecimal: de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b3
3. resulta no binário Base64 para a codificação de texto em ASCII: 3p8sf9JeGzr60+haC9F9mxANtLM=

 É uma forma curta de representar um arquivo.

1. echo “ola mundo” | openssl sha1
2. > (stdin)= de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b3

 Exemplo Prático

1. $ openssl sha1 texto.txt
2. SHA1(texto.txt)= 3e86273b804b9030b096a241d92b84a6ac5f2a35

Fui no arquivo e substituir um (.) por um (,).

Resultado: mudou a string de 40 caracteres

1. $ openssl sha1 texto.txt
2. SHA1(texto.txt)= 2f4575088fffc7018fa88d76175e22e0d83f5a83

 Mudei o arquivo para a forma que estava antes e perceba que a chave voltou a ser a mesma do exemplo 1

1. $ openssl sha1 texto.txt
2. SHA1(texto.txt)= 3e86273b804b9030b096a241d92b84a6ac5f2a35

**Objetos internos do Git**

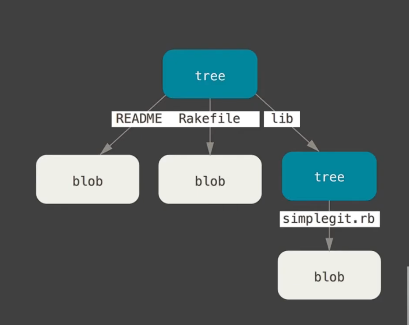
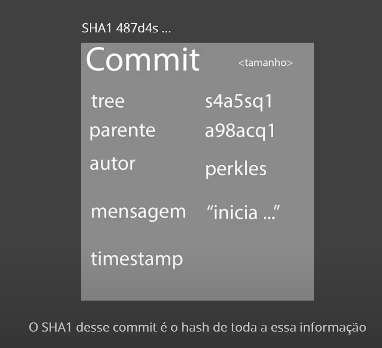
BLOBS = (bolhas) – só guarda o sha daquele arquivo

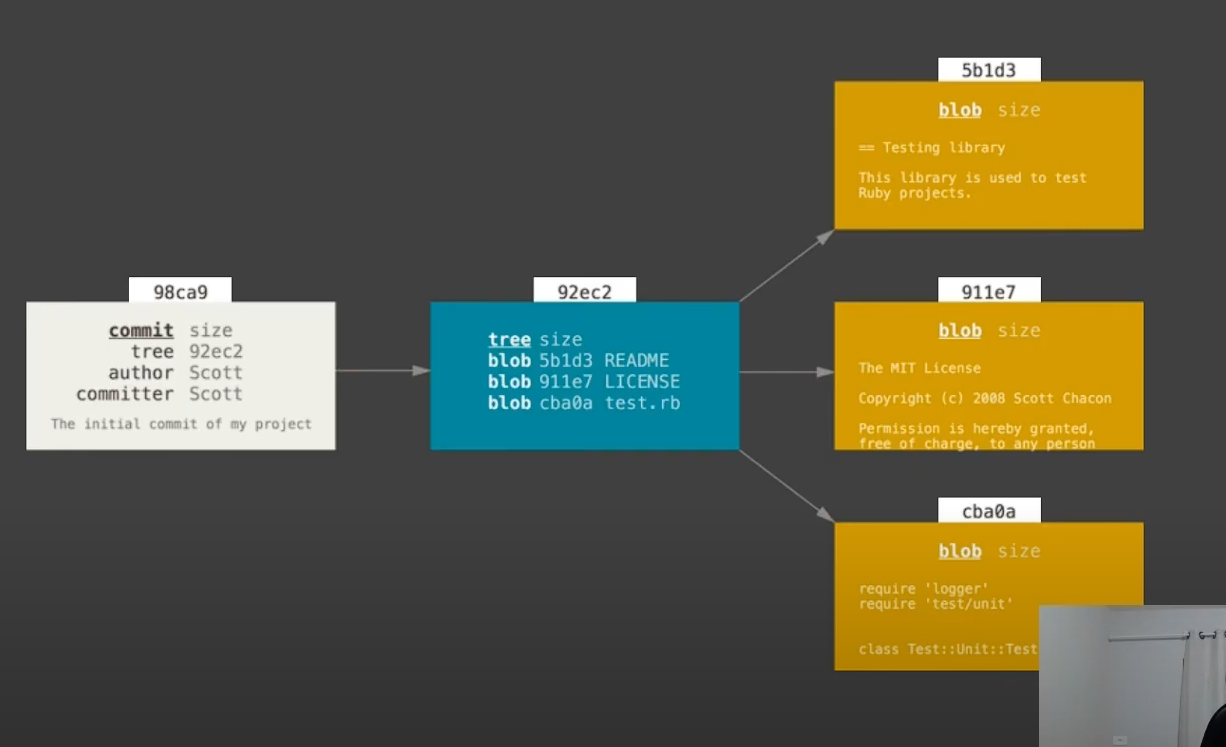
TREES = (arvores) – elas apontam tanto para blobs quanto para outras arvores, tambem guarda o nome do arquivo.

COMMITS = é o objeto que vai juntar tudo, ele aponta para uma arvore, para um parente, para um autor e para uma mensagem também.

Um commit tambem tem um timestamps, este objeto gera a data, a hora de quando ele foi criado.

Os commits tambem possui um **SHA1** .



Sistema distribuído e seguro porque, imagine um projeto onde tem uma equipe de 40 pessoas e cada uma tem a mesma versão do arquivo de forma segura em sua máquina, caso venha acontecer algo com o servidor do git e perder tudo, as 40 maquinas da equipe tem um cópia do mesmo projeto de forma segura e distribuída entre todas as máquinas da equipe.

**Primeiros comandos com o GIT**

- Iniciar o GIT

1. git init

 - Iniciar o versionamento

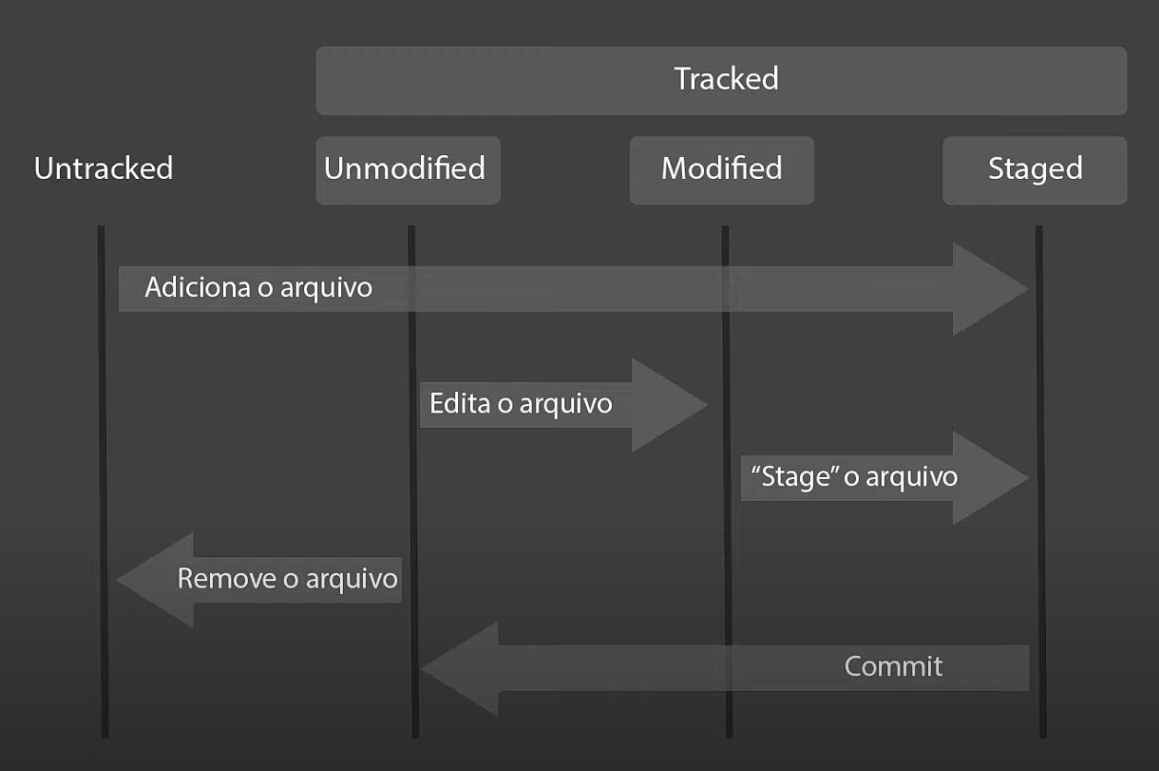
1. git add

 - Criar um commit

1. git commit

Criando um repositório

git init inicializa o repositório dentro da pasta.



Untracked – são os arquivos que o **git**  ainda não tem ciência deles

Tracked – são os arquivos que estão sendo rastreados pelo **git.**

Unmodified – são arquivos que ainda não foi modificados.

Modified – São arquivos que sofreram alterações.

Staged (palco) – é onde ficam os arquivos que ainda estão se preparando para fazer parte de outro tipo de agrupamento, no caso o **commit**.

O **git** acompanha as modificações dos arquivos comparando a chave **SHA1,** toda vez que o arquivo sofre alterações a chave de 40 caracteres muda**.**

Quando executamos o comando **git add**  o arquivo vai para o **stage.**

Se o arquivo não sofrer nenhuma alteração e a gente **deleta,**  o mesmo vai para **untracked.**

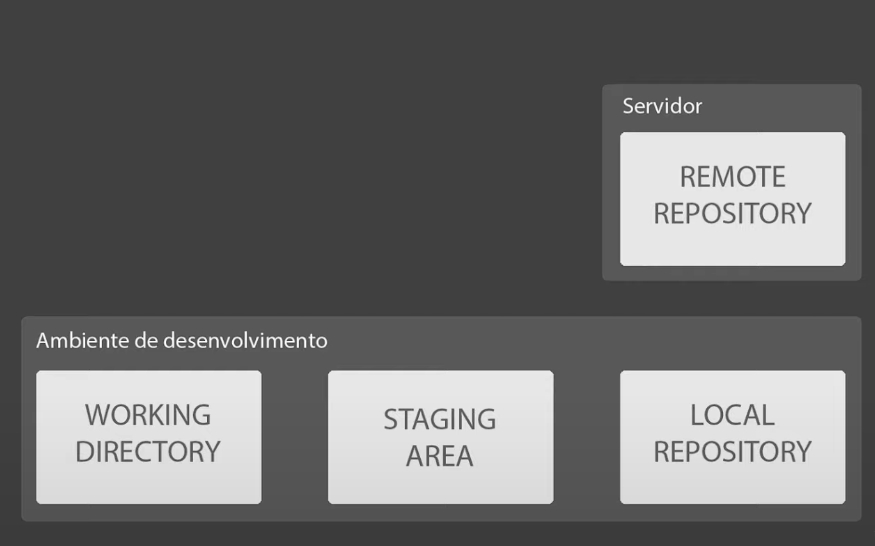
Após o **commit** os arquivos retornam para **unmodified**  para começar o ciclo de novo.

<https://www.devmedia.com.br/principais-caracteristicas-do-git/39634>

**Estados do Git**

Um dos conceitos mais importantes do Git são os estados que podem ser aplicados em cada arquivo. São três estados fundamentais:

* **Commited** – arquivos armazenados na base local.
* **Modified** – arquivos que sofreram mudanças, mas não foram enviados para a base local (commit).
* **Staged** – arquivos modificados marcados para que façam parte do próximo commit.



Tudo que está no repositório local tem que ser comitado, caso contrario você não consegue enviar para o repositório remoto.

**git status** - O comando **git status** exibe as condições do diretório de trabalho e da área de staging. Ele permite que você veja quais alterações foram despreparadas, quais não foram e quais arquivos não estão sendo monitorados pelo **Git**.

1. $ git add strogonoff.md receitas/

 Modifica o estado dos arquivo para **stage**

1. git restore --staged <nome-do-arquivo> //para mudar para unstage
2. $ git add \* // Pega tudo que foi modificado no meu diretório de trabalho e adciona ao stage.

 Podemos usar o git add de várias formas:

1. git add <nome-do-arquivo> // Você especifica quais arquivos vai para **stage**
2. git add \* // Adiciona todos os arquivos modificados ao **stage**
3. git add . // Adiciona todos os arquivos do diretório de trabalho para o **stage**

**Introdução ao GitHub**

1. $ git config --list //Verificar as configurações do git em sua máquina.
2. $ git config --global --unset user.email //Remove meu **email** da configuração.
3. $ git config --global --unset user.nickname //Remove meu **nickname** da configuração.
4. // Terá que configurar novamente o **email** e o **nickname**.
5. $ git config --global user.email [tiagogomes187@gmail.com](mailto:tiagogomes187@gmail.com)
6. $ git config --global user.nickname "Tiago Gomes"

 O GitHub como uma rede social onde podemos curtir, compartilhar e seguir um usuário git.

Por convenção aqueles arquivos que contam a histórias de um repositório é chamado de README.md.

1. $ git remote add origin https://github.com/tiagogomes187/livro-receitas.git

 origin – é apenas um ‘alias’, um apelido, isso para que não precisemos digitar o link remoto o tempo todo.

git push (empurrar) - O comando **git push** é usado para enviar conteúdo do repositório local para um repositório remoto. "**Push**" é como os commits do repositório local são enviados a um repositório remoto. ... As ramificações remotas são configuradas usando o comando **git** remote .

1. git push -u origin main

<https://pt.wikipedia.org/wiki/Git>

**resolvendo conflitos**

**marge -** A mesclagem é o jeito do **Git** de unificar um histórico bifurcado. O comando **git merge** permite que você pegue as linhas de desenvolvimento independentes criadas pelo **git** branch e as integre em um único branch. Observe que todos os comandos apresentados abaixo fazem a mesclagem para o branch atual.

<https://www.atlassian.com/br/git/tutorials/using-branches/git-merge>

**git pull** (puxar) **-** O comando **git pull** é usado para buscar e baixar conteúdo de repositórios remotos e fazer a atualização imediata ao repositório local para que os conteúdos sejam iguais. ... O comando **git pull** é a combinação de dois outros comandos, o **git fetch** , seguido do **git merge** .

1. $ git pull origin main

**git clone –** (clonar) O **git clone** é usado sobretudo para apontar para um repositório existente e fazer um **clone** ou cópia deste repositório no novo diretório, em outro local. O repositório original pode estar localizado no sistema de arquivos local ou em protocolos com suporte a acesso por máquinas remotas.

<https://www.atlassian.com/br/git/tutorials/setting-up-a-repository/git-clone>

