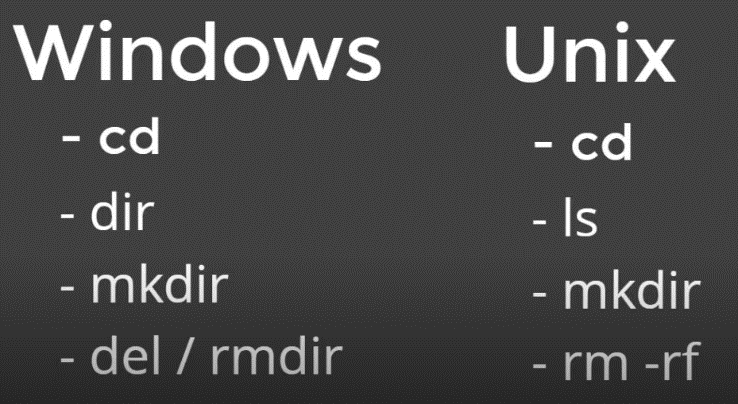
**Introdução ao Git e ao GitHub**



**O Básico do GIT:**

- Mudar de pastas

- Listar pastas

- Criar pastas / arquivos

- Deletar pastas / arquivos

COMANDOS BASICOS

**Entrar em uma pasta**: cd “nome da pasta”

**Voltar um nível de diretório**: cd ..

**Voltar para a raiz (C:\):** cd \

**Limpara a tela do prompt:** cls ou CTRL + L (GIT BASH)

**Auto completar:** cd win (+ tecla TAB)

**Criara diretório/pasta:** mkdir “nome da pasta” – Lembrando que o diretório será criado no nível em que nos encontramos

**Printar uma frase no terminal:** echo - Ex: echo HELLO. Vai retornar HELLO no prompt

**Encaminhar um conteúdo para um arquivo, criando o mesmo:** echo hello > hello.txt

**Deletar o conteúdo de uma pasta:** del “nome da pasta” - Lembrando que será apagado o conteúdo e não o diretório.

**Remover diretório com todo conteúdo:** rmdir “nome do diretório”

**Remover diretório com todo conteúdo sem solicitar confirmação:** rmdir /S /Q

***Funcionamento do GIT – Por baixo dos panos:***

**SHA1:** Significa Secure Hash Algorithm (Algoritmo de Hash Seguro), é um conjunto de funções hash criptográficas projetadas pela NSA (Agência de Segurança Nacional do EUA). Ou seja, é um Algoritmo de encriptação.

Ao rodar o algoritmo em um arquivo, será gerado um conjunto de caracteres identificador de 40 dígitos. Sendo que estes caracteres são sempre únicos para o estado atual de um arquivo.

Quando criamos a chave encriptada com o SHA1, é gerada uma chave única. Se alterarmos uma virgula, é gerada uma nova chave diferente. Caso excluirmos esta virgula e criamos novamente a chave, esta chave será exatamente igual a primeira gerada, auxiliando o controle de versões de arquivo.

**UTILIZANDO O GIT BASH**: Caso queira aplicar o GIT BASH em uma pasta específica, basta clicar com o botão direito do mouse dentro da pasta desejada e clicar na opção de Git Bash Here.

Comando para encriptar dentro do Git Bash: openssl sha1 “nome do arquivo completo”

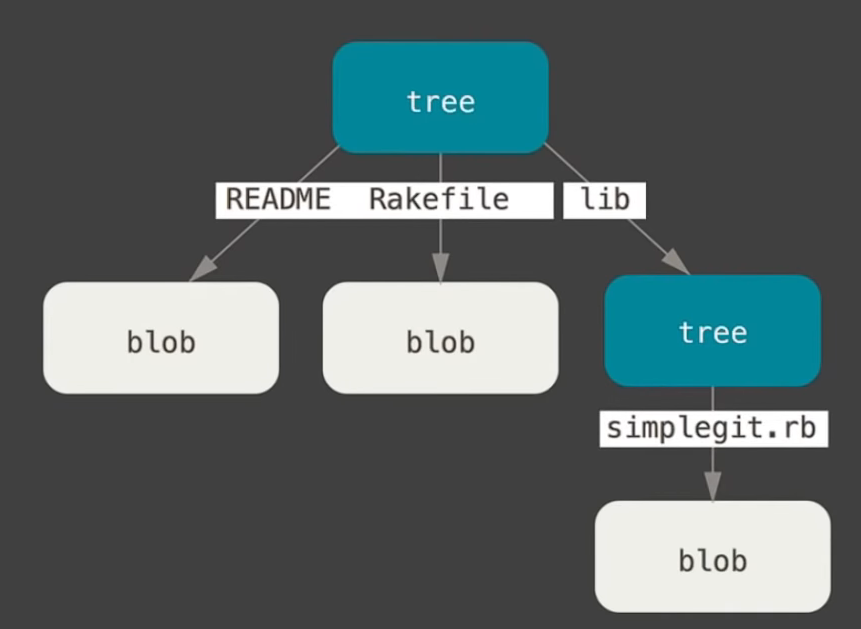
***OBJETOS INTERNOS DO GIT:***

**OS 3 TIPOS BÁSICOS DE OBJETOS DO GIT, RESPONSÁVEIS PELO VERSIONAMENTO DE CÓDIGO:**

echo 'conteudo' | git hash-object --stdin

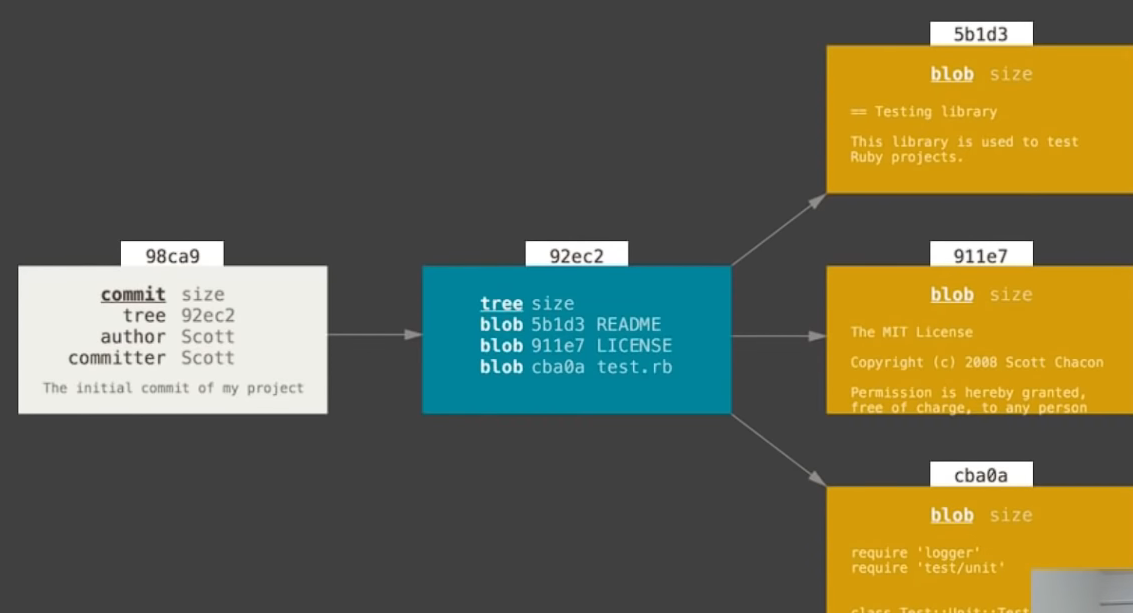
**BLOBS:** Blolha que guarda os metadados do objeto, ou seja, Tipo do objeto, tamanho da string, tamanho do arquivo, entre outros. Disponibiliza o SHA1 do arquivo, mas sem armazenar o nome do mesmo.

**TREES:** Árvores são estruturas que armazenam ou apontam para BLOBS/ARQUIVOS. Também contém metadados dos Blobs É responsável por montar toda a estrutura dos arquivos. Onde estão localizados. As Árvores podem apontar tanto para Blobs quanto para outras arvores (Tree). As Árvores também possuem um SHA1 desse metadado. Se alteramos um arquivo dentro de um Blob, O SHA1 do arquivo será alterado, consequentemente o SHA1 da Bolha e da árvore também serão alterados.

**EX**:

**COMMITS:** É o objeto mais importante, que vai dar sentido para a alteração que está sendo feita. Vai juntar toda a estrutura. O COMMIT aponta para uma Árvore, aponta para um parente (O último Commit realizado antes dele), aponta para um autor, aponta para uma mensagem. A mensagem dentro do commit pode ser escrita para dar sentido a todos os arquivos (Blobs) e estruturas (Trees) dentro deste conglomerado de diretórios. Commits possuem um Timestamp (Carimbo de tempo) determinando para este objeto a data e hora exata em que foi criado/alterado pela última vez.

Importante entender que COMMITS também possuem um SHA1 ou seja, geração de um hash de 40 caracteres identificador do seus metadados. Devido a possuir o parente, é possível “desenhar” uma linha do tempo dos commits realizados.

**EX:**

**CHAVES SSH E TOKENS**:

CHAVE SSH: É uma forma de estabelecer uma conexão segura e encriptada entre duas “máquinas”.

TOKENS DE ACESSO PESSOAL: Tokens gerados para o acesso aos repositórios. Interessante para gerar acesso temporário já que dá para determinar o prazo de validade do mesmo.

**PRIMEIROS COMANDOS COM GIT em Terminal (PROMPT:**

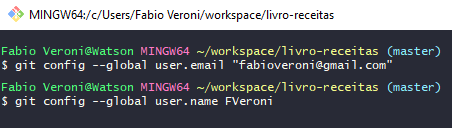
- Iniciar o GIT - git init

- Iniciar o versionamento - git add

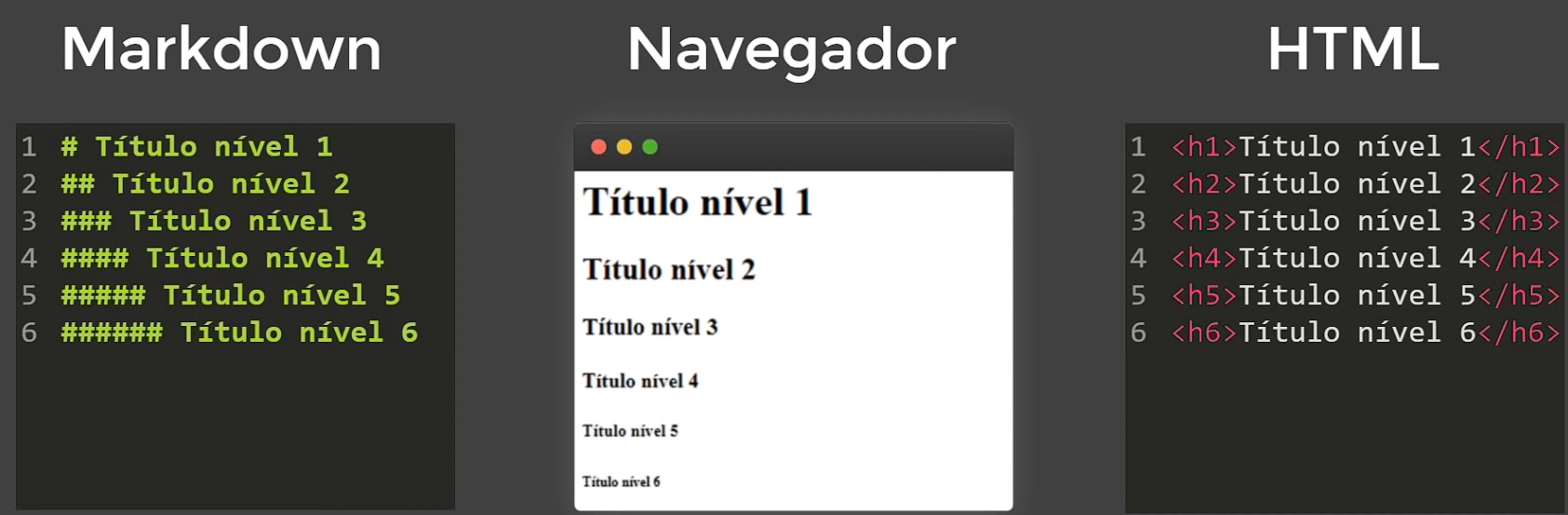
- Criar um commit - git commit

\*COMANDO ls –a (Lista as pastas do diretório – inclusive as ocultas

Comandos para criação de e-mail e user name dos arquivos no GIT



\* Tipo de arquivo Markdown: Uma forma mais Humana de se escrever um arquivo HTML. (Arquivo .mt)



Comandos para comitar:

Git add \* (Para incluir todos os arquivos da pasta no commit)

Git commit –m “Commit Inicial” (Sendo o texto dentro da Aspas o comentário referente ao commit)

\*\* Comando GOT INIT (Criasse um repositório no GIT – Na pasta alocada)

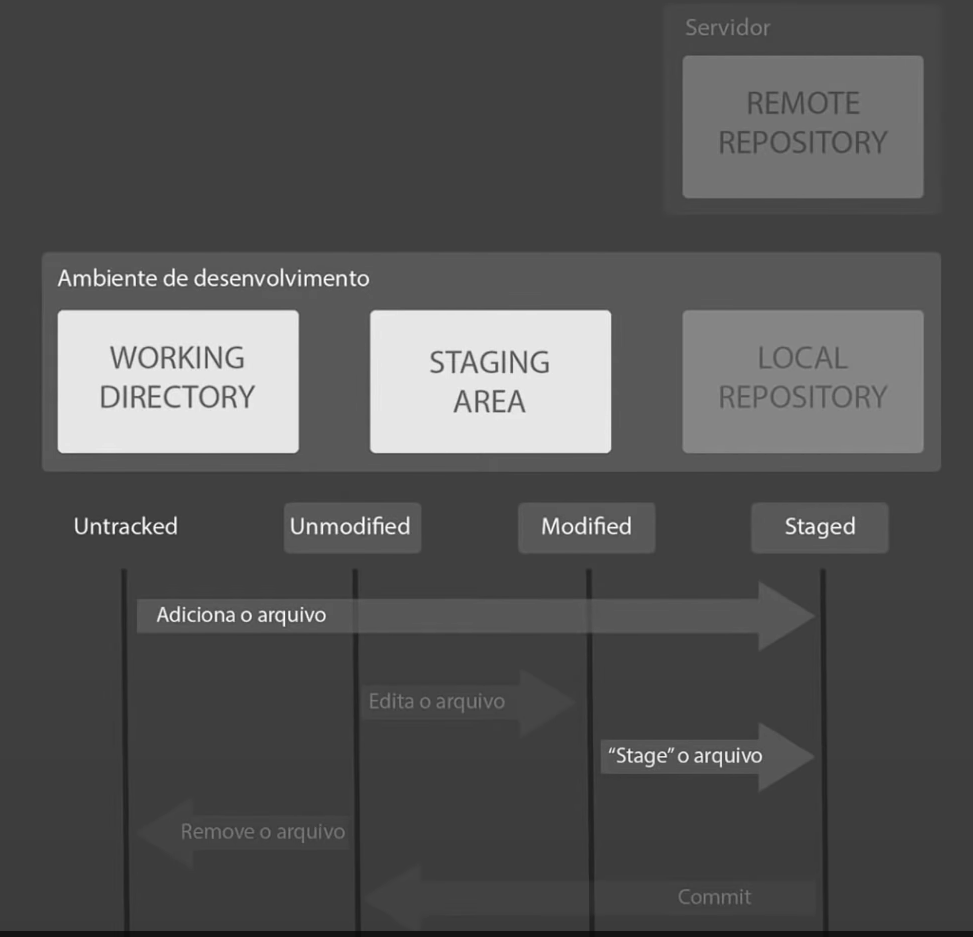
TRACKED e UNTRAKED

UNTRAKED é todo arquivo ainda não alocado ou registrado no GIT.

TRACKED são todos os arquivos que estão alocados ou registrados no GIT.

STAGED são os arquivos que já foram registrados no GIT, foram modificados e aguardam para uma próxima ação que seria o Commit. Após o commit o arquivo volta para o estado Unmodified.

* Estados em repositório local e o repositório remoto.

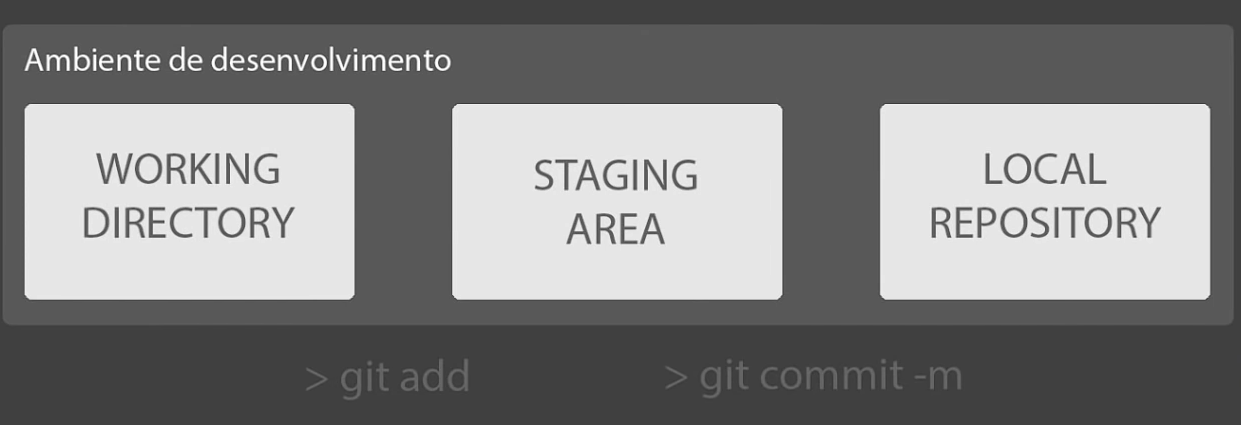


COMANDO ***GIT STATUS***: Informo o status do GIT quanto a arquivos a serem comitados e da árvore de trabalho.

Comando para mover arquivos entre pastas no GIT BASH: mv strogonoff.md ./receitas/

Sendo MV (MOVE) seguido do nome do arquivo e o diretório para qual vamos enviá-lo.

Comando de criação de arquivo direto pelo GIT BASH: echo > README.md <sendo echo o comando e readme.md o nome do arquivo criado>



git config --global --unset user.mail

Comando no GIT BASH para alterar/remover uma configuração Global, onde user.mail é configuração a ser alterada. Para verificar as configurações possíveis, seguir com o comando: git config –list

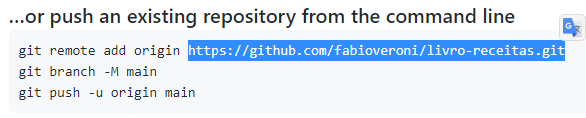
git config --global user.mail "fabioveroni@gmail.com"

**Comando para vinculação da config GLOBAL** de e-mail responsável pela conta sendo nome da função (user.mail) seguido da informação a ser inserida (Email desejado).

***TRABALHANDO COM GITHUB***

Para vincular um repositório no GITHUB para o GIT BASH devemos copiar o link https no Github, entrar no GIT BASH e empurrar para o repositório remoto. Basta executar o comando: **git remote add origin** https://github.com/fabioveroni/livro-receitas.git

Sendo **origin**  um álias para o repositório, não tendo que digitar toda a URL para “empurrar” o repositório localmente para o server remoto



Executando o comando **git remote –v** traremos a lista de repositórios remotos cadastrados

**git push origin master** Comando para empurrar o repositório para o Github

Master no caso acima é a Branch indicada.

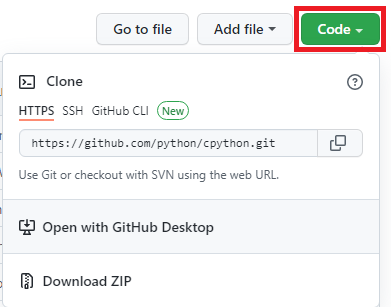
***RESOLVENDO CONFLITOS NO GITHUB***

Comando para puxar repositório alterado no GITHUB com problema de merge

git pull origin máster

Baixando repositório para a área de trabalho local.

Achar o repositório desejado no GITHUB e clicar em **Code**

****

**Copiar o link https**

No GIT BASH entrar no diretório onde desejamos clonar o repositório alvo

Digitar o comando de clone conforme abaixo, sendo o https o copiado

**git clone** https://github.com/python/cpython.git

**COMANDO PARA VERIFICAR REPOSITÓRIOS OCULTOS**: **ls -a**