### **Grupo 2: Artur Leone, Fabricio Almeida, Gabriel Antônio, Luiz Felipe, Luiz Henrique.**

### 

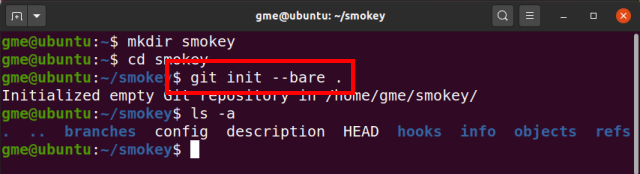
### **Git init**

Para começar um projeto que ainda não seja um repositório (ou repo), o Git Init costuma ser o comando mais indicado.

Basicamente, ele cria um repositório vazio ou transforma uma pasta que você já tem e que não está com controle de versão em um repositório:

git init

Com sua pasta de trabalho devidamente iniciada, é hora de começar a preenchê-la



### **Git clone**

O Git clone é um comando para baixar o código-fonte existente de um repositório remoto (como o Github, por exemplo).

Existem algumas maneiras de baixar o código-fonte, mas é preferível o clone com o modo https:

git clone <https://url-do-link>

Quando você clonar um repositório, o código é copiado para o seu computador e continua linkado ao original, como foi explicado lá na descrição do que é um sistema distribuído.

Se você quiser desvincular a sua cópia do original, rode o comando abaixo:

git remote rm origin

### **Git branch**

Com as *branches* (ou ramificações), vários desenvolvedores podem trabalhar paralelamente no mesmo projeto. Assim, cada um pode codar a sua parte sem que haja confusão!

Considerado um dos comandos Git mais importantes, o branch pode ser usado para três finalidades diferentes: criar, listar e excluir ramificações. Veja, na sequência, como trabalhar com cada uma delas.Para criar uma nova *branch* local, digite:

git branch <nome-da-branch>

Este comando criará uma *branch* local. Para upar a nova *branch* para o repositório remoto, você precisa usar o seguinte comando:

git push -u <remote> <nome-da-branch>

Para ver as ramificações, por sua vez, use:

git branch

ou

git branch --list

Por último, delete uma *branch* a partir do comando*:*

git branch -d <nome-da-branch>

### 

### **Git commit**

Esse comando visa definir um ponto de verificação no processo de desenvolvimento, para o qual você pode voltar mais tarde, se necessário.

git commit -m "mensagem explicando a mudança no código"

### **Git add**

Quando criamos, modificamos ou excluímos um arquivo, essas alterações ocorrerão em nosso ambiente local e não serão incluídas no próximo *commit* (a menos que alteremos as configurações).

Para incluir as alterações de um arquivo em nosso próximo *commit*, será preciso usar o comando git add.

Para adicionar apenas um arquivo:

git add <arquivo>

Para adicionar, de uma vez, todos os arquivos modificados:

git add -A

Vale lembrar que o comando git add não altera o repositório e as alterações não são salvas até usarmos o *git commit*.

**Fonte:** https://blog.geekhunter.com.br/comandos-git-mais-utilizados/