Git

Comandos do Git que você precisa conhecer - Parte 2 - Repositórios Remotos

Conheça comandos e truques do Git que facilitarão o seu trabalho ao lidar com repositórios remotos. Aprenda também a criar um repositório remoto em sua rede local.

Akira Hanashiro • há 4 anos 2 meses

1 - Clonando um Repositório Remoto

Para fazer download de um projeto basta executar o comando `$ git clone`, passando o endereço do repositório. Pode ser Github, Gitlab, Bitbucket, etc.

Esse comando é bem conhecido, mas ele está aqui por um detalhe que muitas pessoas não sabem: por padrão será criada uma pasta com o nome do projeto, mas você também pode no final do comando indicar qual o nome da pasta que você quer que seja criada.

Copiar

```
$ git clone https://meu-endereco.com/meu-projeto.git minha-pasta
```

2 - Repositórios Remotos

Repositórios Remotos são repositórios presentes em outra máquina para a qual nós podemos pegar ou enviar código. Todos esses comandos começam com `$ git remote`. Veja os principais comandos para gerenciá-los:

2.1 - Adicionar Repositórios Remotos

Para ligar o seu repositório local com um repositório remoto, utilize o comando `remote add`. Precisamos passar dois parâmetros para esse comando:

1. **nome**: nome que daremos ao nosso repositório remoto, como se fosse um atalho para não precisarmos ficar escrevendo a URL toda hora. O padrão é usar o nome `origin`
2. **url**: endereço do repositório remoto ao qual o nome passado anteriormente vai se referir

Copiar

```
$ git remote add origin https://meu-endereco.com/meu-projeto.git
```

Caso você queira colocar outro nome ao invés de `origin` não tem problema. Isso pode ser muito útil caso você precise se conectar a mais de um repositório remoto. Mas se tiver apenas um, o recomendado é seguir o padrão e usar o nome `origin`.

Copiar

```
$ git remote -v
```

Teremos um retorno como:

Copiar

```
> origin https://meu-endereco/meu-projeto.git (fetch)
> origin https://meu-endereco/meu-projeto.git (push)
```

2.3 - Remover Repositórios Remotos

Pode ser que você não queira mais o seu local repositório conectado a um repositório remoto. Esse comando não apaga o repositório, apenas desfaz a conexão criada com o comando `$ git remote add`

Copiar

```
$ git remote rm origin
```

Nesse exemplo usamos o nome `origin`, mas lembre-se que ali pode ser o nome que você deu ao seu repositório remoto. Se você deu um nome diferente de `origin`, lembre-se de usar o comando anterior para listar os repositórios remotos.

2.4 - Renomear Repositórios Remotos

Pode ser que você não queira mais o nome que você deu ao seu repositório remoto com o comando `$ git remote add`. Há um comando bem simples para renomear.

Copiar

```
$ git remote rename nome-atual novo-nome
```

2.5 - Alterar Endereço de Repositórios Remotos

Já escreveu o endereço de um repositório remoto errado ou migrou ele para outro serviço? Com o comando `set-url` você será capaz de apenas alterar o endereço sem precisar mexer em mais

```
$ git remote set-url origin http://meu-novo-endereco/meu-projeto.git
```

3 - Branches Remotas

3.1 - Listando Branches

Esse comando lista todas as branches presentes no repositório do seu computador.

Copiar

```
$ git branch
```

Caso você queira que ele liste também as branches que estão no repositório remoto, adicione `-a` :

Copiar

```
$ git branch -a
```

3.2 - Criando Branches Remotas

Ao enviar o seu código para uma branch remota que ainda não existe, basta executar o `push` com a opção `-u` junto com o nome do repositório remoto e o nome da nova branch.

Copiar

```
$ git push -u origin minha-branch
```

Após a branch remota estar criada, você poderá executar simplesmente `$ git push` .

Também podemos escrever assim:

Copiar

```
$ git push origin :minha-branch
```

3.4 - Renomeando Branches Remotas

Vimos no post anterior que para renomear uma branch local executamos:

Copiar

```
$ git branch -m nome-atual novo-nome
```

Após renomear a sua branch local, basta apagar a branch remota com o nome antigo e fazer um push com a branch com o novo nome:

Copiar

```
$ git push origin :nome-atual novo-nome
```

Para terminar de ligar a branch local com a remota, entre na branch com o novo nome e execute:

Copiar

```
$ git push origin -u novo-nome
```

4 - Achando o culpado

Deu problema e estão dizendo que foi você? Não mais! Com o comando `blame` você pode ver quem alterou cada linha de um arquivo. (a menos que realmente tenha sido você 😂)

Copiar

```
git blame nome-do-arquivo
```

E ainda podemos usar `-e` para que seja exibido o email ao invés do nome do usuário.

5 - Compartilhamento Offline / Local

Podemos ter repositórios remotos que não estão na nuvem, como Github, Gitlab, ou Bitbucket. Podemos fazer com que um computador em nossa rede seja o responsável por armazenar o código e receber o *push* de todos os usuários. Isso é útil principalmente em empresas que possuem uma política mais rígida com a segurança de suas informações e não permitem que seus códigos sejam armazenados fora de seus próprios servidores.

Outro uso para isso é quando adicionamos mais de um repositório remoto. Você pode continuar fazendo commits para o Github e ao mesmo tempo ir armazenando uma cópia em um pendrive.

Primeiro, no local onde teremos o nosso repositório "servidor" (que receberá os pushes), iniciamos um repositório junto com `--bare`. Neste repositório você não poderá editar arquivos ou fazer commits, ele só permite receber pushes.

Copiar

```
$ git init --bare
```

Se você iniciou, por exemplo, um repositório em uma pasta dentro de um pendrive com o caminho `E:\meu-projeto`, basta clonar com o comando:

Copiar

```
$ git clone E:\meu-projeto
```

Ou então, como já vimos, podemos adicionar esse repositório na nossa lista de repositórios remotos, permitindo que nossos commits sejam enviados para algum lugar como o Github e para o pendrive ao mesmo tempo:

Copiar

```
$ git remote add usb E:\meu-projeto
```

Nesse exemplo demos o nome `usb` ao invés de `origin`. Lembre-se que você pode dar qualquer nome.