Git Criando Pasta

Crie uma pasta em um diretório de sua escolha
Abra essa pasta
Clique com o botão direito

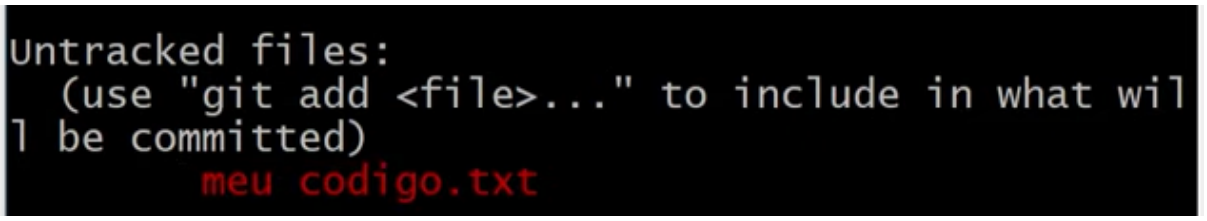
Selecione: `Git Bash Here` - Isso abrirá uma linha de comando na pasta

Agora digite os comandos:

```
Git init - cria pasta oculta do git
```

Adicionando Arquivos ao Controle de Versão:

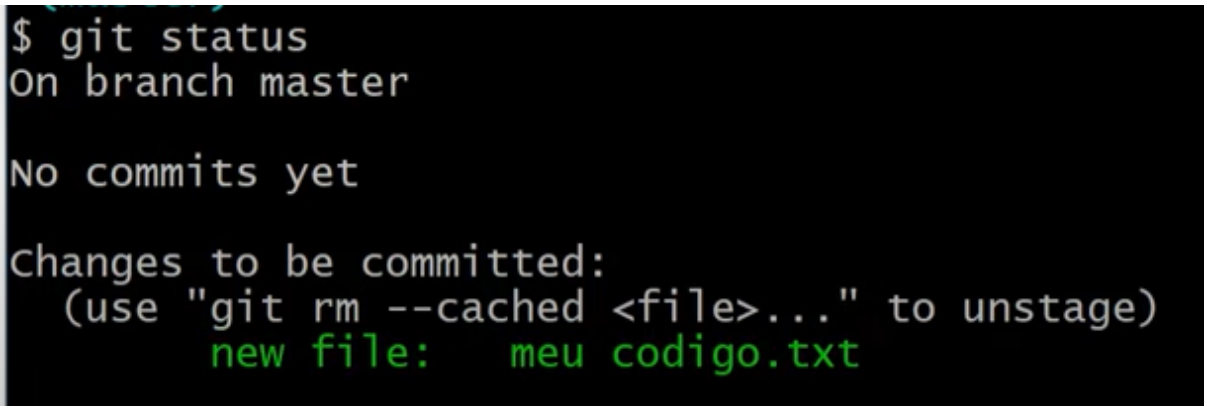
Caso você adicione um arquivo de nome "meu codigo" na pasta, você pode verificar se ele está no controle de versão pelo comando: `git status`



Caso não esteja no git, aparecerá o arquivo em vermelho. Para adicionar o arquivo ao controle de versão deve-se adicionar o comando:

```
git add "meu codigo.txt"
```

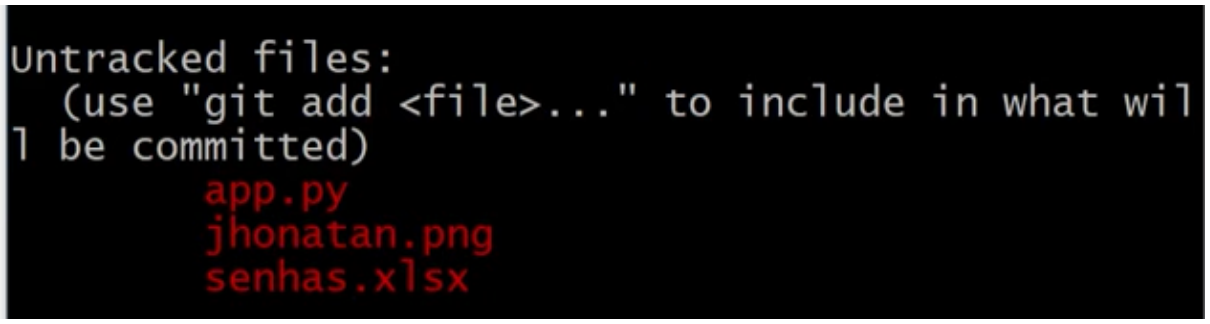
Caso novamente digite o comando git status verá que o arquivo está no controle de versão:



Para o caso que haja mais de um arquivo esse não é o caminho adequado.

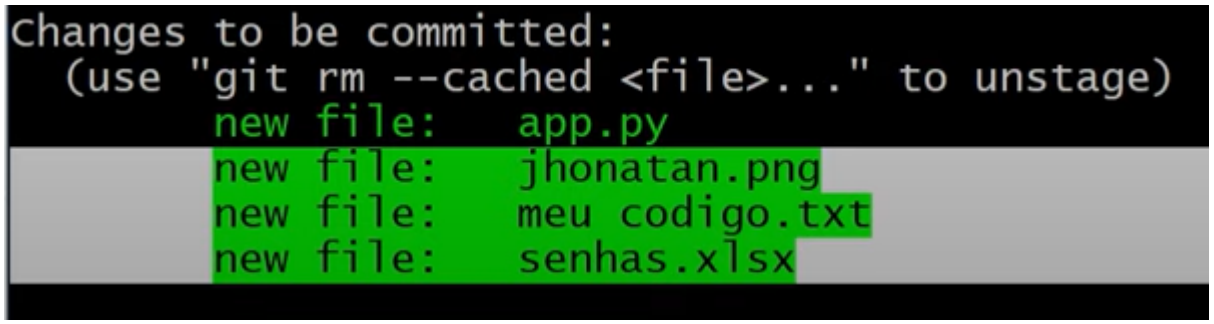
Por exemplo os arquivos:

Que ainda não estão no controle de versão:



Para os adicionar deve-se usar o comando: `git add .`

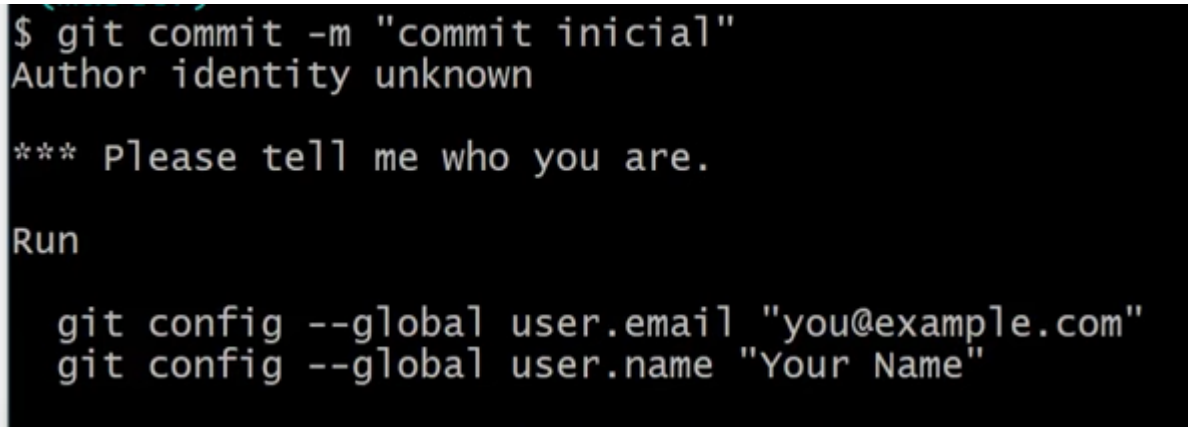
Com o git status podemos verificar que os arquivos estão agora no controle de versão:



Adicionando Arquivos na nuvem:

É válido ressaltar que esses arquivos ainda não estão na nuvem do github. Para isso é necessário fazer o comando: `git commit -m "initial commit"` => o -m é para a mensagem que estará entre aspas duplas onde você dirá o que há nessa versão. Apenas por padrão, o primeiro commit sempre é chamando de commit initial.

Quando feito esse comando pela primeira vez é pedido que o git seja configurado no pc:



Basta repetir esse comando e colocar seu email e seu nome de usuário do github.

Por fim, para enviar para a nuvem: `git push`

No entanto, é necessário criar um repositório no github para onde serão enviados os arquivos.

Assim o comando é:

```
git remote add origin endereçodorepositorio
```

Definido agora para onde enviar o código dê o comando: `git push --set-upstream origin master`

Assim o código será enviado para o brench master.

Atualização de código:

Caso o código tenha atualizações e você deseja verificar se foi enviado para o git, basta rodar novamente o comando git status. Caso não esteja, basta fazer o processo anterior:

```
git add .
git commit -m "oquefoimodificado"
git git push
```

Verificar o histórico do código e voltar à versão anterior:

```
git reflog
```

A versão mais atual é a que está no topo.

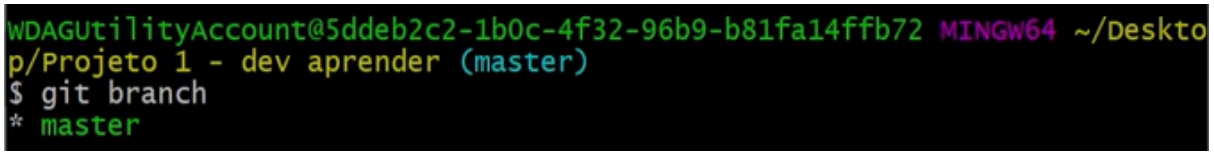
Em amarelo abaixo do git reflog tem a id do código mais recente e, logo abaixo, do mais antigo. Para voltar, para uma dada versão, basta digitar uma dessas id. A exemplo, voltar para a 1658aa4:

```
git reset --hard 1658aa4
```

Trabalhando com Branches:

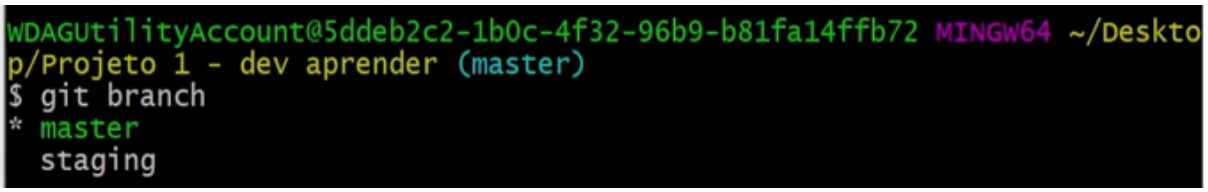
Para verificar as branches que existem usa-se o comando:

```
git branch
```



Para então criar uma, usa-se o mesmo comando e define-se um nome:

```
git branch staging => esse nome é dado entre programadores para branches que recebem atualizações ainda não testadas.
```



Em asterisco está a branch que está sendo trabalhada no momento (no caso a master). Para mudar de branch usa-se o comando:

```
git checkout staging => vai mudar para a branch staging
```

```
WDAGUtilityAccount@5ddeb2c2-1b0c-4f32-96b9-b81fa14ffb72 MINGW64 ~/Deskto
p/Projeto 1 - dev aprender (staging)
$ git branch
  master
* staging
```

Ao criar uma branch nova, deseja-se enviar para a nuvem. Assim, o comando - que normalmente é sugerido pelo git - é:

```
git push --set-upstream origin staging
```

Unindo Branches:

Para unir a branche com a branche principal - o master - deve-se usar o código:

Entrar na branch que vai receber as atualizações:

```
git checkout master
```

Unir as branches:

```
git merge staging
git push
```

Antes de fazer merge, deve-se fazer o push das atualizações que estão no servidor para a sua máquina para garantir que você irá unir os códigos nas versões mais atuais possíveis:

```
git pull
```

Os passos então são:

1. git pull da branch principal
2. gerar uma nova branch a partir da branch principal
3. Trabalhar e adicionar novas funcionalidades na nova branch que criou
4. Finalizar o trabalho na branch temporária
5. Git checkout na branch principal
6. Git pull
7. Mergiar(unir) o código da branch temporário com a branch principal(depois de testar)
8. Git push da branch principal

Git Ignore:

Arquivos que não se deseja enviar para o controle de versão, mas que estão na pasta. Para fazer isso, deve-se utilizar o comando git ignore pelo próprio terminal do git. O comando é:

```
touch .gitignore
```

O comando abrirá um txt no qual você deve por o nome dos arquivos/pastas que você deseja ignorar. O nome dos arquivos/pastas nesse txt devem conter uma barra - sem espaço - ao final do nome. Normalmente é feito na branch master.