SENAI

Serviço Nacional de Aprendizagem Industrial

PELO FUTURO DO TRABALHO

Aula 01 - Github: Avançando no Git e GitHub

Professor: Rafael Ventura

SENAI
PELO FUTURO DO TRABALHO

O que iremos aprender:

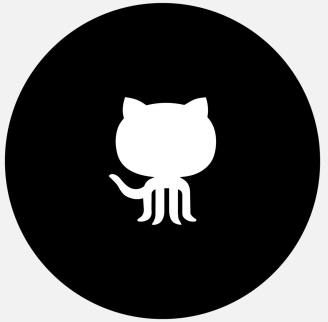
- Voltando no tempo com o git log
- Revertendo um commit
- Resetando um commit
- Alterando um commit
- Criando um readme
- Trabalhando com branches
- Merge e Rebase
- Salvando para depois
- Comparando commits diferentes
- Releases e tags
- Criando um perfil no seu github
- Criando o arquivo git ignore

Referências

- Git: https://git-scm.com/doc
- GitHub: https://docs.github.com/pt
- Markdown: https://wordpress.com/support/markdown-quick-reference/

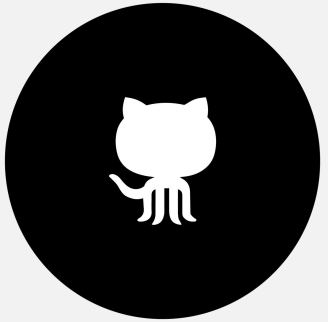
Voltando no tempo ...

```
git log
```



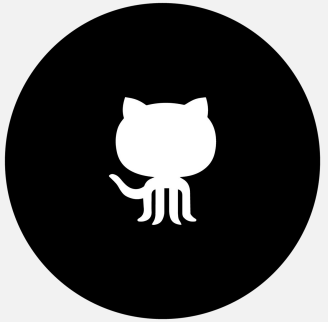
Revertendo um commit

```
git revert <hash-do-commit>
```



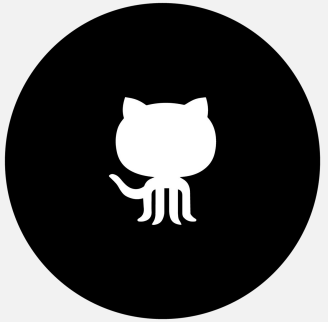
Confirmando a reversão ...

```
git log
```



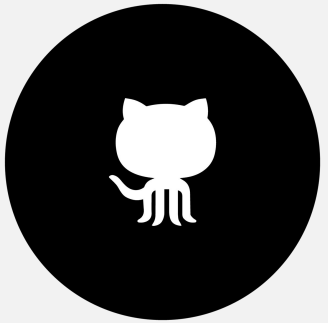
Resetando um commit ...

```
git reset --hard <hash-do-commit-anterior>
```



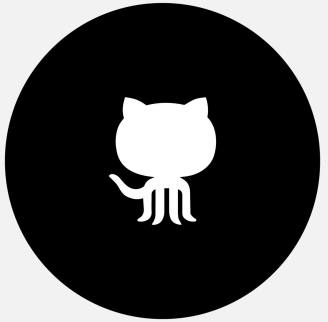
Resetando um commit ...

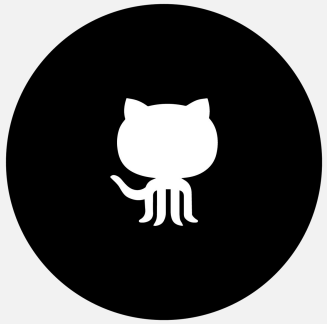
- Como boa prática, evite usar o comando reset depois de ter enviado as alterações para o repositório remoto.
- Enquanto as alterações estiverem no repositório local o uso do reset é mais aconselhável.



Restaurando antes de um commit ...

```
git restore --staged <nome-do-arquivo>
```

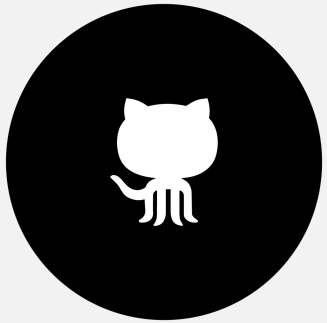




Alterando o último commit

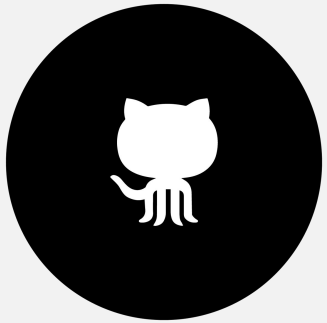
```
git commit --amend -m “commit-corrigido”
```

```
git commit --amend <nome-do-arquivo>
```



Criando um README.md

- Basta você criar um arquivo README.md em seu repositório e começar a editá-lo.
- Uma boa ideia é acelerar essa produção usando o CHAT GPT.
- Markdown é uma linguagem de marcação parecida com HTML, ela aceita todas as tags HTML.

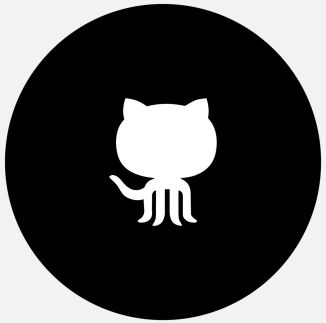


Criando um perfil com um README.md

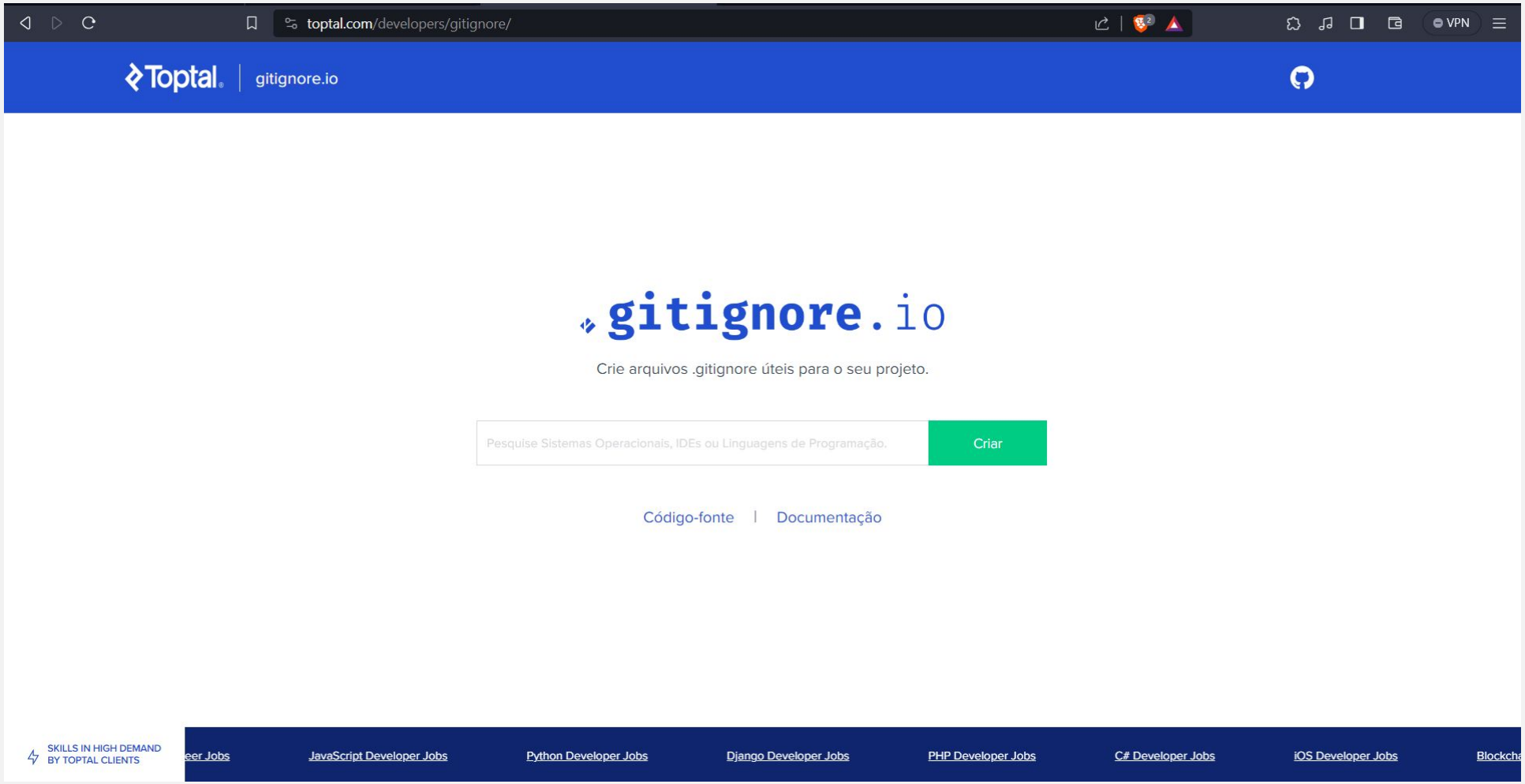
- Crie um novo repositório.
- Coloque o nome do repositório como o seu nome de usuário GitHub.
- Selecione a opção PUBLIC.
- Adicione um Readme.
- Edite agora seu Readme.

Ignorando arquivos ou pastas do seu repositório local

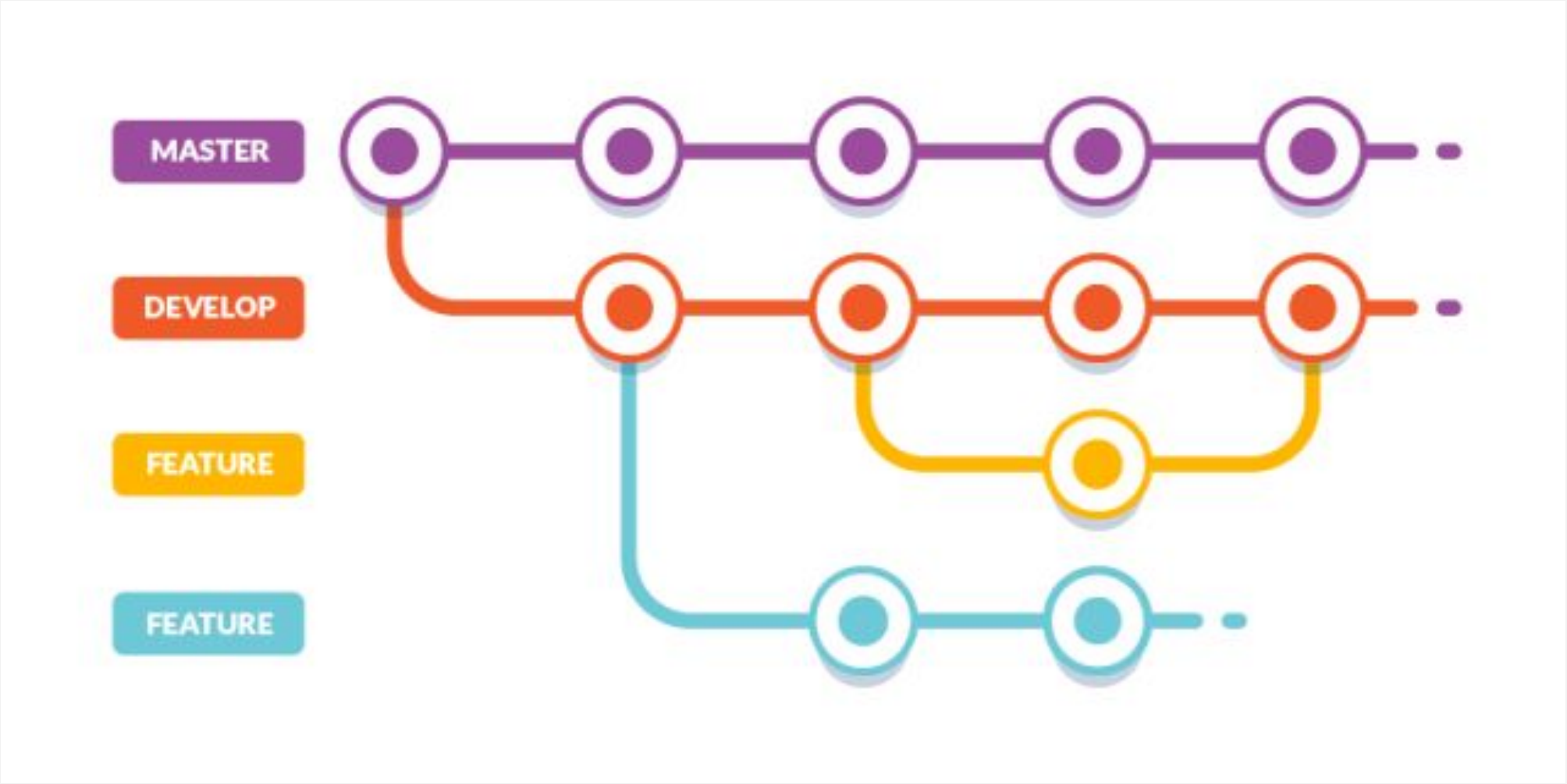
- Crie um arquivo chamado .gitignore
- Edite o .gitignore passando:
- /exemplo (ignora todo o conteúdo da pasta exemplo)
- arquivo.txt (ignora o arquivo.txt)
- *.css (ignora qualquer arquivo que tenha a extensão css)

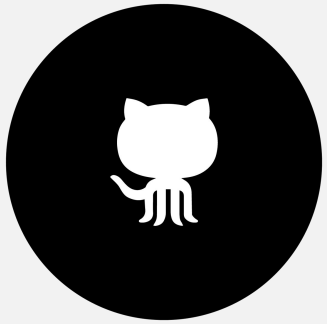


Criando um .gitignore de forma rápida



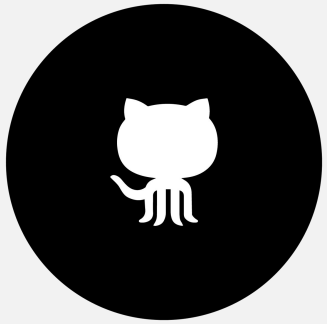
Branches





Criando Branches ...

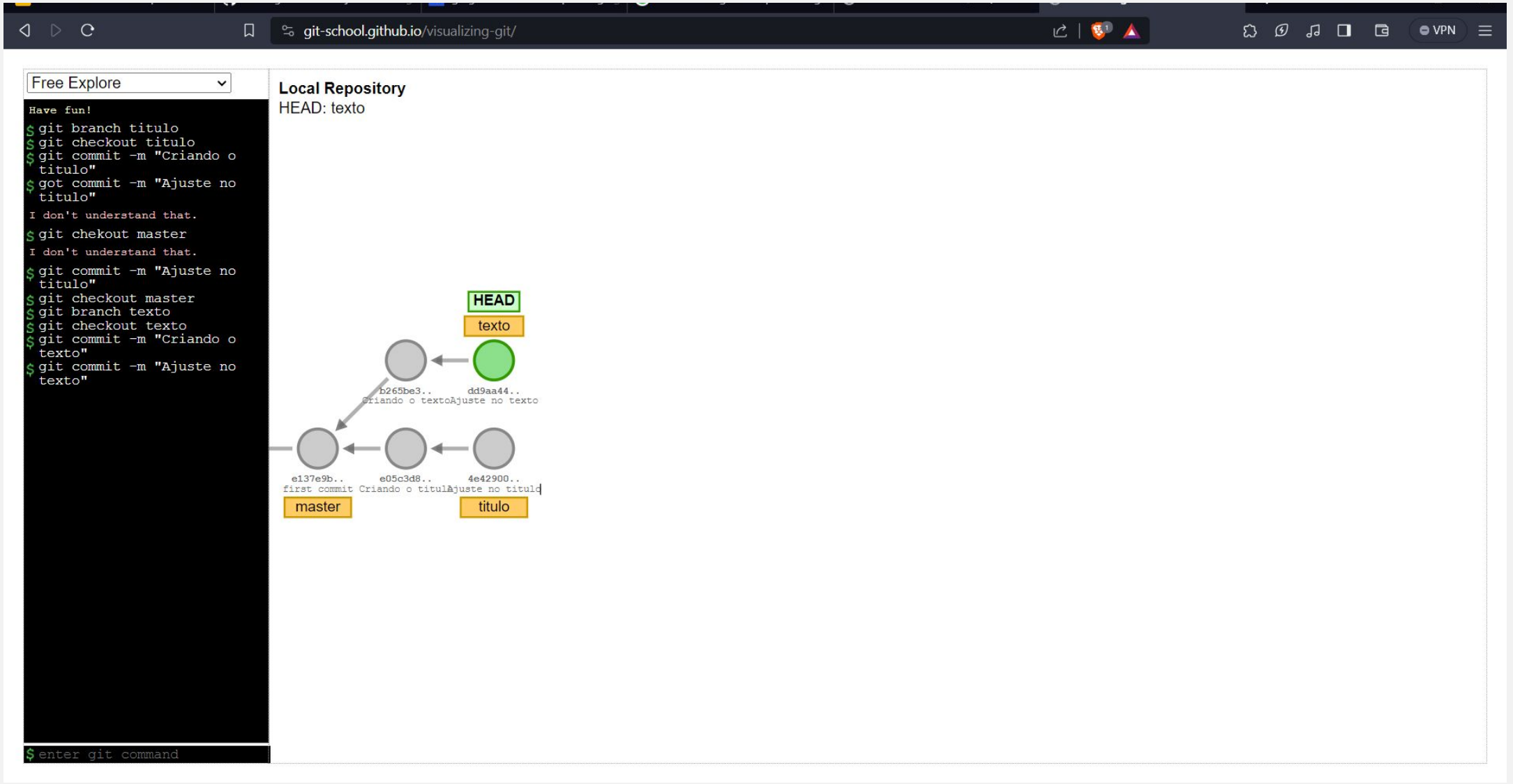
```
git branch <nome-da-branch>
```

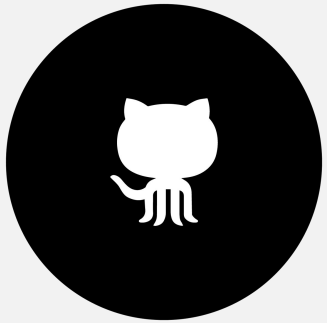


Mudando de Branches ...

```
git checkout <nome-da-branch>
```

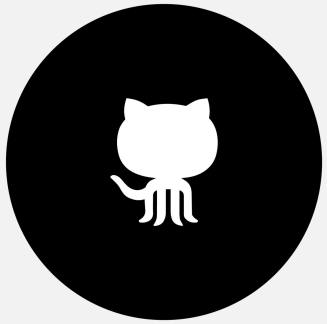
Visualizando as Branches





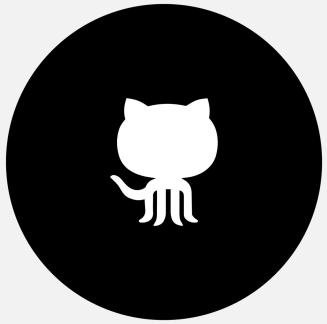
Exercício de Branches

- Crie duas nova branches.
- Faça alterações nos arquivos.
- Faça um commit
- Utilize o comando git log para visualizar as mudanças
- Retorne a branch master



Atalho para criar branches

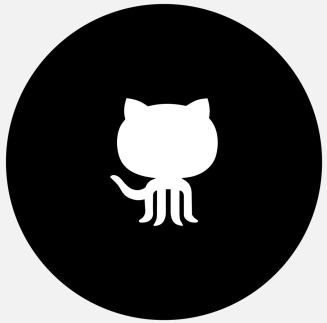
```
git checkout -b <nome-da-branch>
```



Unindo o commits de branches separadas (merge)

```
git checkout <nome-da-branch-recebe-merge>
```

```
git merge <nome-da-branch-envia-merge>
```



Exercícios de Merge

- Crie duas nova branches.
- Faça alterações nos arquivos.
- Faça um commit
- Faça um checkout para a branch master
- Execute o comando git merge <nome-da-branch-alterada>
- Execute o git log

Visualizando o Merge

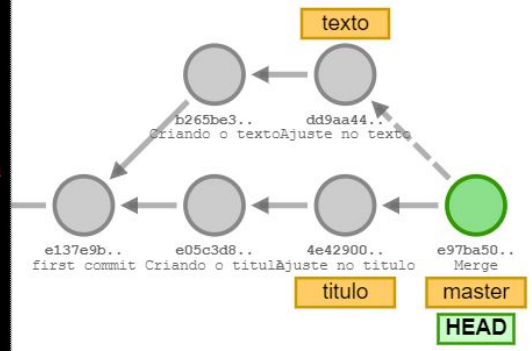
git-school.github.io/visualizing-git/

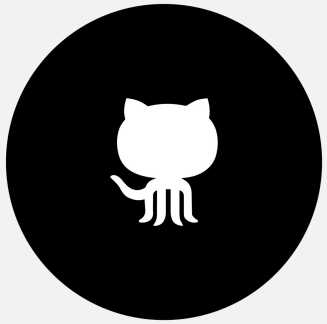
Free Explore

```
Have fun!
$ git branch titulo
$ git checkout titulo
$ git commit -m "Criando o
  titulo"
$ got commit -m "Ajuste no
  titulo"
I don't understand that.
$ git chekout master
I don't understand that.
$ git commit -m "Ajuste no
  titulo"
$ git checkout master
$ git branch texto
$ git checkout texto
$ git commit -m "Criando o
  texto"
$ git commit -m "Ajuste no
  texto"
$ git checkout master
$ git merge titulo
You have performed a fast-forward
merge.
$ git checkout master
$ git merge texto
```

$ enter git command

Local Repository
HEAD: master



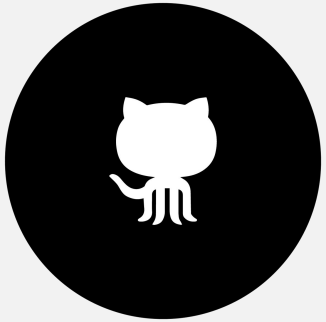


Visualizando o log graficamente

```
git log --graph
```

Atualizando a Branch sem Merge

```
git rebase <nome-da-branch-atualizada>
```



Visualizando o Rebase

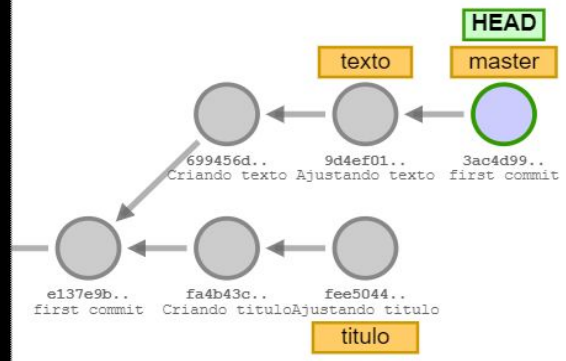
git-school.github.io/visualizing-git/

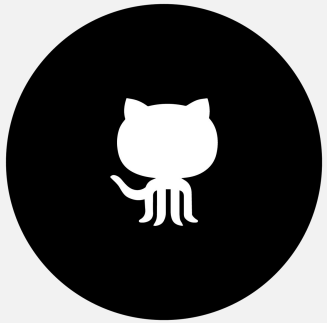
Free Explore

```
Have fun!
$ git checkout -b titulo
$ git commit -m "Criando
  titulo"
$ git commit -m "Ajustando
  titulo"
$ git checkout master
$ git checkout -b texto
$ git commit -m "Criando
  texto"
$ git commit -m "Ajustando
  texto"
$ git checkout master
$ git rebase texto
```

$ enter git command

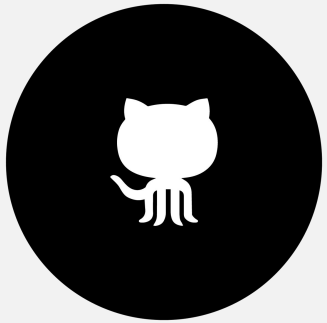
Local Repository
HEAD: master





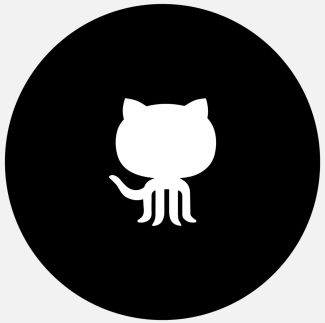
Merge/Rebase

- Merge integra o conteúdo da branch de trabalho com a branch master
- Apenas a branch master é alterada para adicionar as mudanças e o histórico da branch de trabalho permanece inalterado.
- O Rebase move a base da branch de trabalho para o final da branch master.
- O código também é integrado porém ele faz isso transformando duas branches em uma só. A linha do tempo permanece como uma linha contínua.



Exercícios de Conflitos

- Gere um conflito.
- Crie uma branch separada da master
- Faça um commit nesta branch
- Retorne para a master e altere o mesmo arquivo na mesma linha
- Execute o comando git merge <nome-da-branch-alterada> ou git rebase <nome-da-branch-alterada>
- Execute o git log –graph

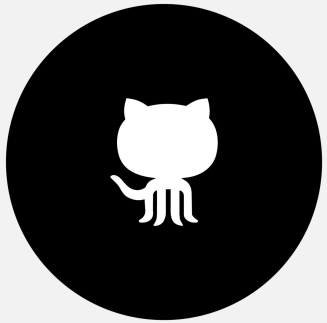


Salvando para depois ...

```
git stash
```

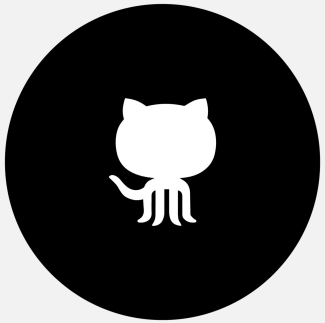
```
git stash list
```

```
git stash pop
```



Exercícios de Stash

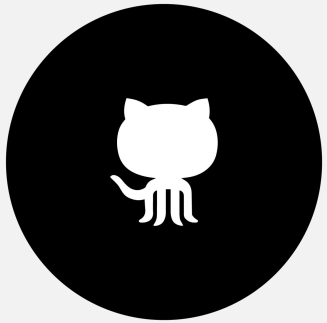
- Faça uma alteração qualquer
- Salve essa alteração com o git stash.
- Execute o git stash list para confirmar o save.
- Faça uma nova alteração
- Faça um commit dessa nova alteração
- Execute o git stash pop para trazer a alteração salva.



Uma outra forma de viajar no tempo...

```
git log --oneline
```

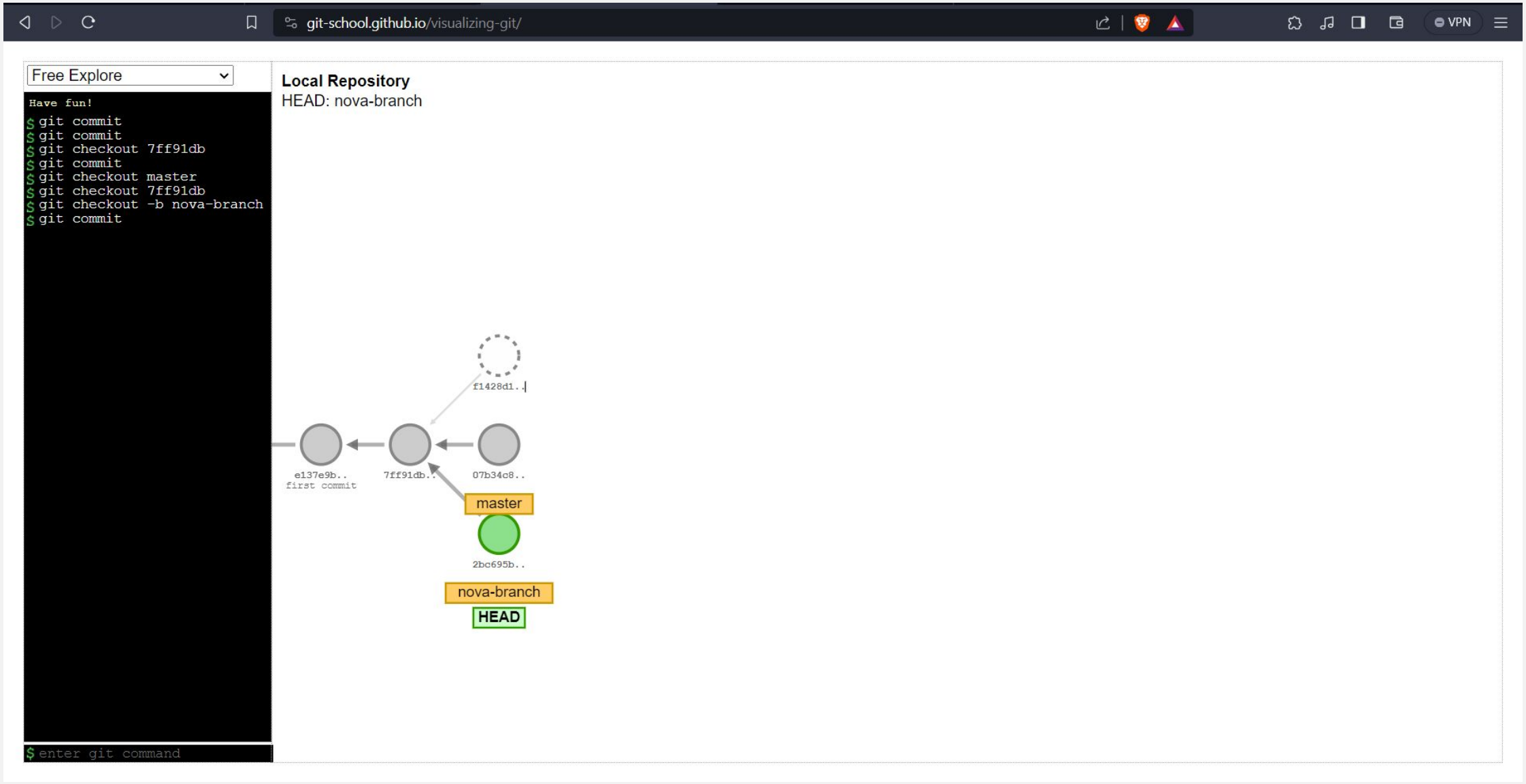
```
git checkout <hash-do-commit>
```

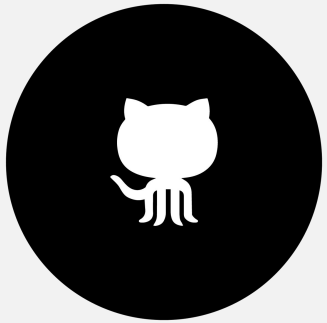


Viajando com checkout

- Quando passamos o hash com o comando checkout
- Voltamos na linha do tempo para uma branch desanexada a branch HEAD(local)
- Qualquer alteração agora só será guardada caso criarmos uma nova branch

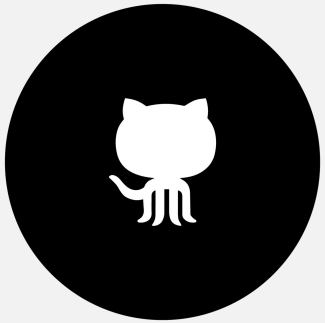
Visualizando o checkout





Exercícios com checkout

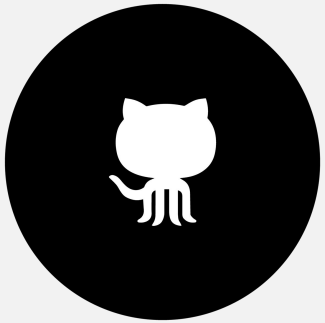
- Faça uma alteração qualquer
- Execute o git log para viajar no tempo.
- Execute o comando git checkout <hash-do-commit>
- Faça mais uma alteração comitando
- Crie uma nova branch
- Faça um novo commit



Comparando as alterações

```
git diff
```

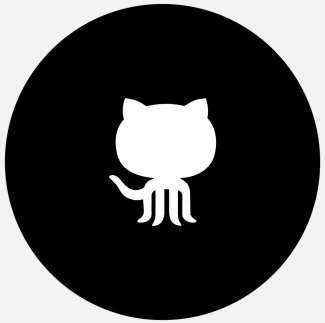
```
git diff <hash-do-commit>..<hash-do-commit>
```



Gerando as entregas (criando versões)

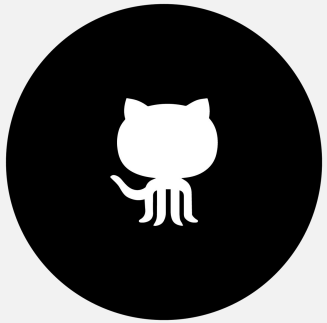
```
git tag -a v1.0.0 -m “mensagem”
```

```
git tag
```



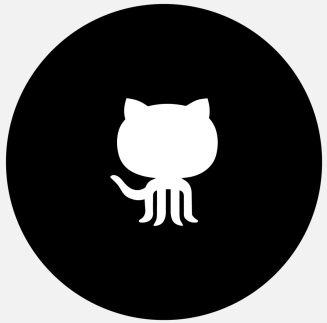
Enviando sua tag para repositório remoto

```
git push origin v1.0.0 main
```



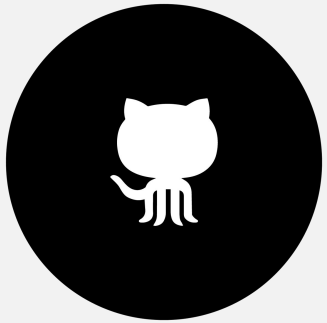
Tags

- Marco temporal importante em seu projeto.
- Geralmente utilizada para criar versões do código.
- Uma tag sinaliza que aquele código não será mais alterado.
- Qualquer alteração neste código deve ser feito em uma nova versão.



Releases

- Versão específica do seu projeto que é disponibilizada para download ou deploy
- Empacota um conjunto de arquivos do seu repositório em um formato que possa ser facilmente distribuído.
- Ao criar uma Release no GitHub, você pode escolher quais commits, Tags ou branches deseja incluir no pacote.



Exercícios Releases

- Crie uma release no GitHub
- Envie uma tag para o seu repositório remoto
- Na página sobre tags clique no botão criar nova release
- Adicione as tags que você gostaria que sua release tenha.

SENAI

Serviço Nacional de Aprendizagem Industrial

PELO FUTURO DO TRABALHO

0800 048 1212 **sc.senai.br**

Rodovia Admar Gonzaga, 2765 - Itacorubi - 88034-001 - Florianópolis, SC