

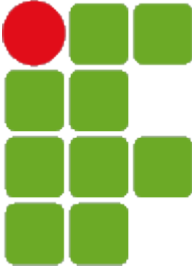
Instituto Federal de Santa Catarina
CST em Análise e Desenvolvimento de Sistemas
Introdução à Computação

Controle de versão, Git e GitHub

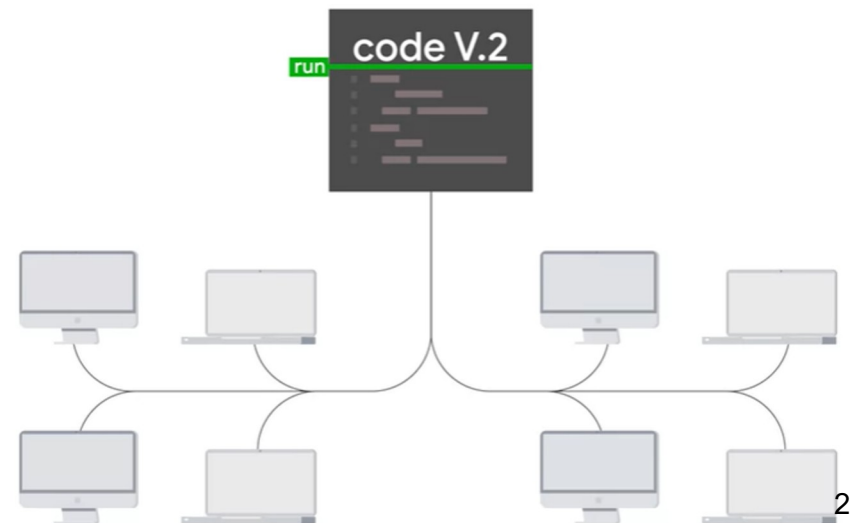
Prof. Diego da Silva de Medeiros

São José, abril de 2024

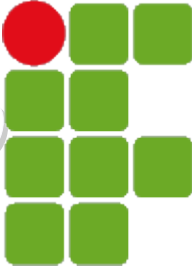
Antes do controle de versão



- Você adicionou uma nova funcionalidade numa script que checa computadores
- Ao instalar em todos os computadores, metade dos computadores acusam problema
- Seu código precisa ser atualizado
 - Opção 1: Você pode alterar rapidamente o código, e mandar aos computadores, mas alterações rápidas causam mais bugs
 - Então depois da primeira atualização, você precisa fazer uma segunda, terceira, etc
 - Opção 2: Retornar o código a uma versão anterior à funcionalidade, e corrigir com tempo o problema causado



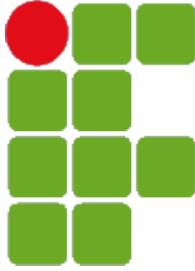
Forma primitiva do controle de versão



- Várias pessoas trabalhando no desenvolvimento, cada um com uma parte do código
- Adiciona uma funcionalidade, e guarda o arquivo como nova versão
- Muito manual:
 - Lembrar de fazer a cópia
 - Precisa fazer a cópia de tudo, mesmo que tenha modificado somente uma parte
 - Mandando por e-mail para os colegas, quem fez o que? Porque fizeram?
- Como poderia ser?



Diferenciando arquivos e aplicando alterações



- Duas cópias do mesmo código, como diferenciar?
 - Comando DIFF
 - Saída pode ser armazenada num arquivo *changes.diff*
- E para aplicar alterações do arquivo *.diff*?

```
$ cat menu1.txt
Menu1:

Apples
Bananas
Oranges
Pears

$ cat menu2.txt
Menu:

Apples
Bananas
Grapes
Strawberries

$ diff -u menu1.txt menu2.txt
--- menu1.txt      2019-12-16 18:46:13.794879924 +0900
+++ menu2.txt      2019-12-16 18:46:42.090995670 +0900
@@ -1,6 +1,6 @@
-Menu1:
+Menu:

 Apples
 Bananas
-Oranges
-Pears
+Grapes
+Strawberries
```

```
$ cat hello_world.txt
Hello World

$ cat hello_world_long.txt
Hello World

It's a wonderful day!

~$ diff -u hello_world.txt hello_world_long.txt
--- hello_world.txt      2019-12-16 19:24:12.556102821 +0900
+++ hello_world_long.txt  2019-12-16 19:24:38.944207773 +0900
@@ -1 +1,3 @@
 Hello World
+
+It's a wonderful day!

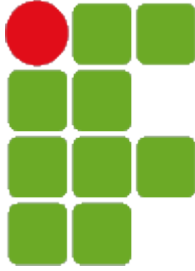
$ diff -u hello_world.txt hello_world_long.txt > hello_world.diff

$ patch hello_world.txt < hello_world.diff
patching file hello_world.txt

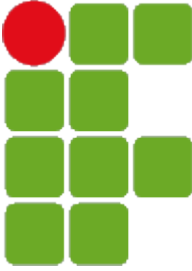
$ cat hello_world.txt
Hello World

It's a wonderful day!
```

Version Control System (VCS)

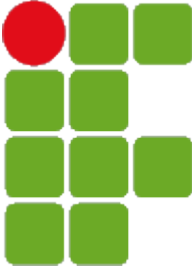


- Sistema que mantém um registro das alterações que fazemos nos arquivos
- Sabemos quais alterações foram feitas, quando, quem, porque, etc
- Podemos reverter uma alteração realizada se for necessário
- Facilita a colaboração, por permitir mesclar (**merge**) alterações de fontes diferentes
- Múltiplas edições de múltiplos arquivos podem ser tratados como uma alteração simples, chamada de **commit**
- Pode armazenar mais do que somente código: arquivos de configuração, documentação, arquivos de dados, etc
- Pode ser valioso mesmo para um único desenvolvedor:
 - Armazena histórico de alterações, trazendo insights sobre decisões do passado
- Quando uma alteração causa problema, pode ser facilmente desfeita
- Para programas em servidores com redundância, garante que todas as máquinas estão com a mesma versão
- **Stack Overflow**: mais de 90% dos devs usam Git em seus projetos



- VCS criado em 2005 por Linus Torvalds (sim, o mesmo do Linux)
- Criado para gerenciar o desenvolvimento do kernel do Linux, que usava desenvolvedores do mundo todo
- **Arquitetura distribuída:** todo colaborador possui uma cópia inteira do repositório em sua máquina
- Pode funcionar:
 - Máquina sem acesso à internet (standalone)
 - Como um servidor que hospeda o repositório
 - Como um cliente que acessa o repositório
- Alternativas: Subversion, Mercurial, etc
- Documentação do Git: <https://git-scm.com/doc>
- Sobre controle de versões: https://en.wikipedia.org/wiki/Version_control

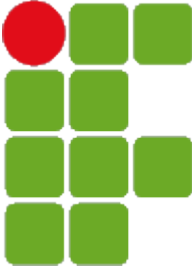
3 áreas do Git



- **Git directory** (pasta .git): database do projeto, que guarda as mudanças nos arquivos e o histórico de alterações (**snapshot**)
- **Working tree**: área fora da pasta .git, que tem a versão atual do projeto, e onde fazemos alterações nos arquivos
- **Staging area (index)**: área que contém as alterações marcadas para serem incluídas no próximo **commit**



Usando o Git



```
# Configurando o usuário, que será usado para registrar alterações
$ git config --global user.email "me@example.com"
$ git config --global user.name "My name"

# Criando um novo repositório vazio no diretório atual ou reinicializa um repositório existente
$ mkdir ~/Pasta
$ cd ~/Pasta
$ git init
Initialized empty Git repository in ~/Pasta/.git/

(master)$ ls -la
total 4
drwxr-xr-x 1 Diego 197121 0 Nov 29 15:10 ./
drwxr-xr-x 1 Diego 197121 0 Nov 29 15:10 ../
drwxr-xr-x 1 Diego 197121 0 Nov 29 15:10 .git/

# Criando um arquivo e registrando sua alteração (mandar para Staging Area - index)
(master)$ echo "#! /bin/bash" > script.sh

(master)$ git add script.sh

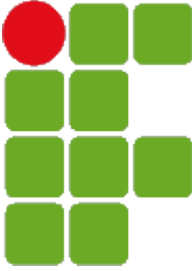
# Verificando informações sobre a Working Tree atual e alterações pendentes
(master)$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   script.sh

# Efetivando mudanças (snapshot)
(master)$ git commit # Adicionar um comentário indicativo das alterações
```


Arquivo no Git



- Arquivos podem fazer parte dos snapshots (**tracked**) ou não (**untracked**, normalmente para arquivos novos)
- Arquivos podem assumir um dos 3 estados:
 - **Modified**
 - Arquivo **tracked** pelo Git, que sofreu alguma modificação
 - Modificação realizada por qualquer editor de textos (vi, nano, Code, atom)
 - **Staged**
 - Modificações foram marcadas para registro
 - Uso do comando `git add <arquivo>`
 - **Committed**
 - Modificações foram armazenadas no VCS

Usando o Git

Verificando o estado atual

```
(master)$ git status
```

On branch master

nothing to commit, working tree clean

Criando e editando um arquivo novo (**modified**)

```
(master)$ Code arquivo_teste.sh
```

```
(master)$ git status
```

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: arquivo_teste.sh

no changes added to commit (use "git add" and/or "git commit -a")

Adicionando o arquivo na **staging area**

```
(master)$ git add arquivo_teste.sh
```

```
(master)$ git status
```

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: arquivo_teste.sh

Salvando o snapshot (**committed**)

```
(master)$ git commit -m 'Add ":" ao fim da mensagem.' # Detalhe para o comentário no próprio comando
```

[master f4d9378] Add ":" ao fim da mensagem.

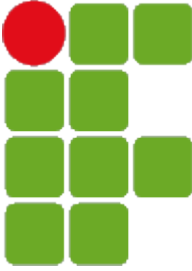
1 file changed, 1 insertion(+), 1 deletion(-)

```
(master)$ git status
```

On branch master

nothing to commit, working tree clean

Boa mensagem de commit



- Normalmente é quebrada em algumas seções
 - Primeira linha: sumário curto, seguido por uma linha em branco
 - Próximas linhas: descrição completa das alterações
- Nunca use mensagens pouco informativas
- Histórico de mensagens pode ser visto com o comando `git log`

```
# Verificando o histórico de commits
```

```
(master)$ git log
commit 5f5facaa5268f74025ae1d09c3d48a7cf09eb8db (HEAD -> master)
Author: Diego Medeiros <diegomedeiros@ifsc.edu.br>
Date:   Wed Nov 29 17:52:18 2023 -0300
```

Adicionando a opção `-a` ao `LS`

O programa não estava encontrando os arquivos ocultos, por isso foi necessário adicionar a opção `-a` ao `ls`.

```
commit 85b8a3320d72343cfca6bb57bdcb3e28bd7a0ceb
Author: Diego Medeiros <diegomedeiros@ifsc.edu.br>
Date:   Wed Nov 29 17:49:50 2023 -0300
```

Criando o arquivo

Esta script irá resolver todos os problemas do mundo.

Mais alguns comandos úteis

```
# Atalho para colocar qualquer arquivo alterado na staging area e realizar o commit  
(master)$ git commit -a
```

```
# Histórico mais completo, trazendo as alterações realizadas
```

```
(master)$ git log -p  
Author: Diego Medeiros <diegomedeiros@ifsc.edu.br>  
Date: Thu Nov 30 23:30:04 2023 -0300
```

Mensagem muito bem escrita sobre as alterações realizadas

```
diff --git a/script.sh b/script.sh  
new file mode 100644  
index 0000000..36ac368  
--- /dev/null  
+++ b/script.sh  
@@ -0,0 +1 @@  
+#! /bin/bash
```

```
# Mostra as alterações realizadas num commit <hash-id>
```

```
(master)$ git show <hash-id>
```

```
# Mostra alterações atualmente realizadas, para saber o que foi feito antes de fazer um commit
```

```
(master)$ echo "ls -l" >> script.sh  
(master)$ git diff  
diff --git a/script.sh b/script.sh  
index 36ac368..39bd292 100644  
--- a/script.sh  
+++ b/script.sh  
@@ -1 +1,2 @@  
  #! /bin/bash  
+ls -l
```

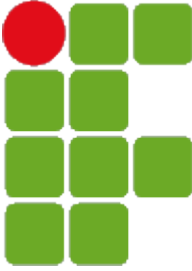
```
# Deleta ou move (ou renomeia) um arquivo
```

```
(master)$ git rm <arquivo>  
(master)$ git mv <origem> <destino>
```

```
# Alterar editor de textos padrão do GIT
```

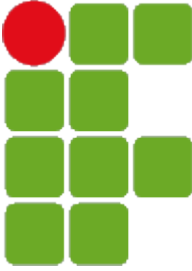
```
(master)$ git config --global core.editor "nano"
```

.gitignore



- Arquivo usado para dizer ao Git para ignorar alguns arquivos num repositório
 - Útil para arquivos do sistema operacional
- Usa um formato específico para passar os nomes dos arquivos
- Pode ser criado no GitHub (veremos mais depois)
- Mais informações em <https://git-scm.com/docs/gitignore>
- Mais comandos em <https://education.github.com/git-cheat-sheet-education.pdf>

Desfazendo alterações



- Uma das características mais poderosas de VCSs
- Executada com o comando `git checkout <arquivo>`

```
(master)$ cat script.sh
#!/bin/bash

(master)$ git status
On branch master
nothing to commit, working tree clean

# Realizando uma alteração no arquivo de código
(master)$ echo "ls -l" >> script.sh
(master)$ cat script.sh
#!/bin/bash
ls -l

# Alteração ainda não foi colocada na staging area
(master)$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   script.sh

no changes added to commit (use "git add" and/or "git commit -a")

# Cancelando a alteração
(master)$ git checkout script.sh
Updated 1 path from the index

(master)$ cat script.sh
#!/bin/bash
```

Desfazendo alterações na staging area

```
# Realizando uma alteração no arquivo de código
(master)$ echo "ls -l" >> script.sh

# Enviando à staging area
(master)$ git add script.sh

# Verificando que a alteração está registrada
(master)$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   script.sh

# Tentando desfazer a alteração
(master)$ git checkout script.sh
Updated 0 paths from the index

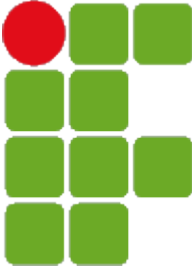
# Mas observe que ela ainda está lá
(master)$ cat script.sh
#!/bin/bash
ls -l

# Retirando o arquivo script.h da staging area
(master)$ git reset HEAD script.sh
Unstaged changes after reset:
M    script.sh

# Observe que agora é possível usar o checkout para desfazer a alteração
(master)$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   script.sh

no changes added to commit (use "git add" and/or "git commit -a")
```

Corrigindo um commit



- Esqueceu de adicionar um arquivo, ou a mensagem do commit está ruim?
- Um commit pode ser sobrescrito com o comando `git commit --amend`
- Deve ser evitado caso o commit já tenha sido publicado num repositório compartilhado, pois esconde o histórico de alterações


```

# Criando duas nova scripts
(master)$ touch script1.sh script2.sh

# Enviando apenas uma das duas scripts à staging area, um erro
(master)$ git add script1.sh

# Criando um commit com o erro
(master)$ git commit -m 'Adicionando 2 novas scripts'
[master d81401c] Adicionando 2 novas scripts
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 script1.sh

(master)$ git log
commit f7b0e83832be838ea081a99f4967f17cf1f7b93b (HEAD -> master)
Author: Diego Medeiros <diegomedeiros@ifsc.edu.br>
Date:   Fri Dec 1 00:19:00 2023 -0300

    Adicionando 2 novas scripts

# Observe que a script2.sh está marcada como untracked
(master)$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    script2.sh

nothing added to commit but untracked files present (use "git add" to track)

# Enviando a segunda script à staging area
(master)$ git add script2.sh

# Segunda script está pronta para ser registrada
(master)$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   script2.sh

# Realizando um amend, para corrigir o erro do registro da script2.sh
(master)$ git commit --amend # O editor de textos abre esperando um comentário
[master 5c18727] Adicionando 2 novas scripts
Date: Fri Dec 1 00:19:00 2023 -0300
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 script1.sh
create mode 100644 script2.sh

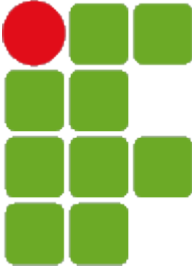
# Veja que o amend sobrescreve o commit anterior
(master)$ git log
commit 5c18727ecd2bd1b25eca1cc8d0e64b8c153e2052 (HEAD -> master)
Author: Diego Medeiros <diegomedeiros@ifsc.edu.br>
Date:   Fri Dec 1 00:19:00 2023 -0300

```

Adicionando 2 novas scripts

Incluí um AMEND porque eu esqueci a segunda script.

Rollbacks



- Um **commit** é executado com um trecho de código com erro
- É possível reverter a última alteração, fazendo um **commit** inverso, com o comando **git revert HEAD**
- Não apaga o histórico
- **Commits** mais antigos podem ser restaurados com **git revert <hash-id>**

```
# Adiciona um novo comando ao script
(master)$ echo pwdddddd >> script.sh

# Salva o snapshot com a alteração
(master)$ git commit -a -m 'Adiciona o comando PWD'
[master 2178c72] Adiciona o comando PWD
1 file changed, 1 insertion(+)

# Observando o registro do commit
(master)$ git log
commit 2178c725df8f990355c209db8024e4dd8c932bf2 (HEAD -> master)
Author: Diego Medeiros <diegomedeiros@ifsc.edu.br>
Date:   Fri Dec 1 00:37:25 2023 -0300
```

Adiciona o comando PWD

```
# Tenta rodar a script para ver se está tudo certo (fazer antes do commit!!)
(master)$ ./script.sh
total 1
-rwxr-xr-x 1 Diego 197121 29 Dec  1 00:36 script.sh
-rw-r--r-- 1 Diego 197121  0 Dec  1 00:15 script1.sh
-rw-r--r-- 1 Diego 197121  0 Dec  1 00:15 script2.sh
./script.sh: line 3: pwdddddd: command not found
```

Rollbacks

```
# Revertendo o commit anterior, criando um novo commit,  
# só que com as linhas inversas do realizado anteriormente  
(master)$ git revert HEAD          # Editor de textos abre para adicionar algum comentário ao revert  
[master 0252c3a] Revert "Adiciona o comando PWD"  
1 file changed, 1 deletion(-)
```

```
# Observando a alteração realizada no log detalhado  
(master)$ git log -p -2  
commit 0252c3aeb34fbd3d56f85dc0ad86a4f91e32830c (HEAD -> master)  
Author: Diego Medeiros <diegomedeiros@ifsc.edu.br>  
Date:   Fri Dec 1 00:37:59 2023 -0300
```

```
Revert "Adiciona o comando PWD"  
O comando estava errado  
This reverts commit 2178c725df8f990355c209db8024e4dd8c932bf2.
```

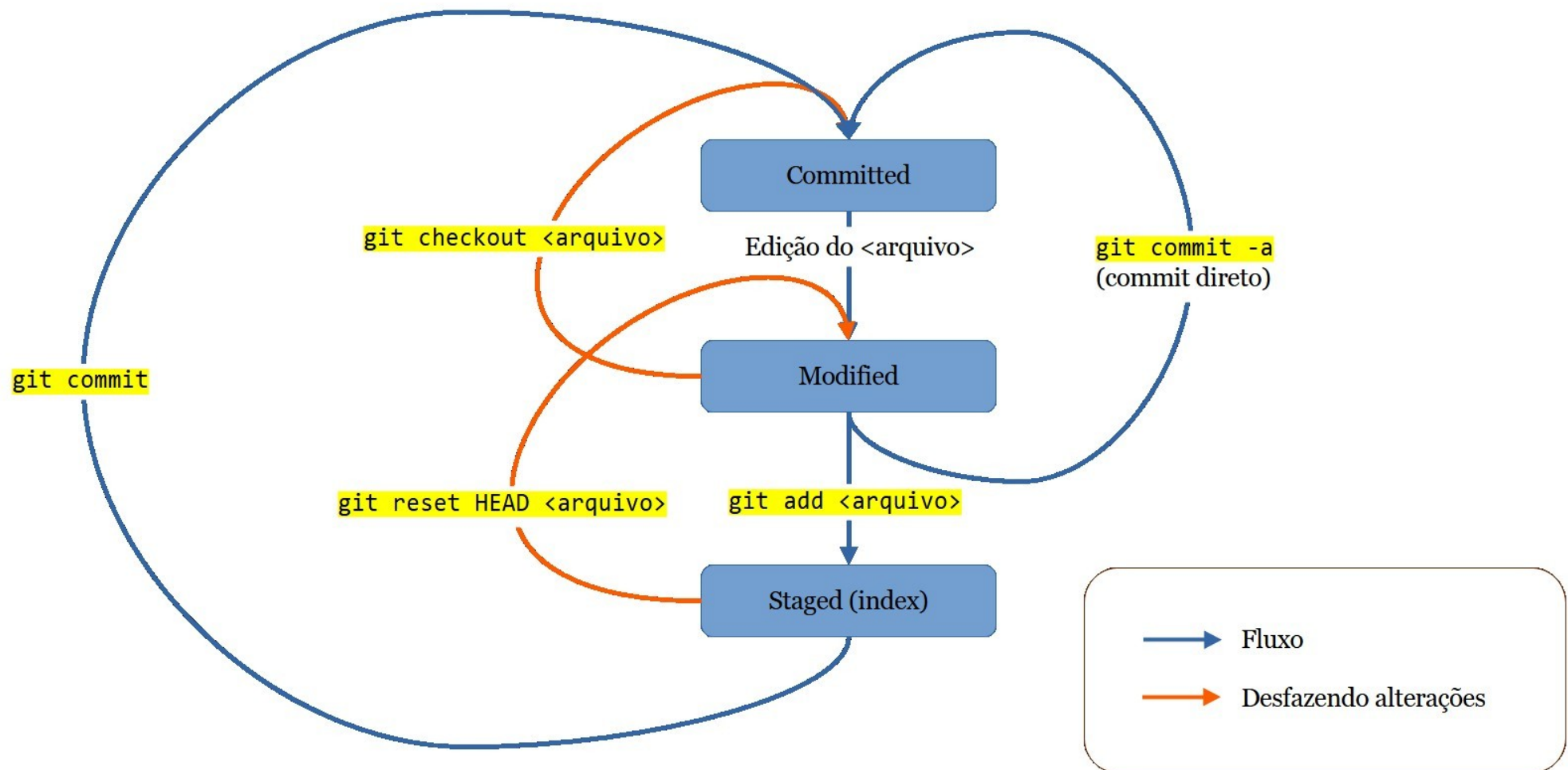
```
diff --git a/script.sh b/script.sh  
index 0827242..39bd292 100644  
--- a/script.sh  
+++ b/script.sh  
@@ -1,3 +1,2 @@  
#! /bin/bash  
ls -l  
-pwdddddd
```

```
commit 2178c725df8f990355c209db8024e4dd8c932bf2  
Author: Diego Medeiros <diegomedeiros@ifsc.edu.br>  
Date:   Fri Dec 1 00:37:25 2023 -0300
```

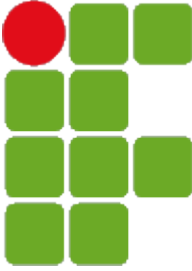
```
Adiciona o comando PWD
```

```
diff --git a/script.sh b/script.sh  
index 39bd292..0827242 100644  
--- a/script.sh  
+++ b/script.sh  
@@ -1,2 +1,3 @@  
#! /bin/bash  
ls -l  
+pwdddddd
```

Diagrama (local)

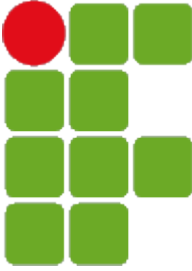


Exercício

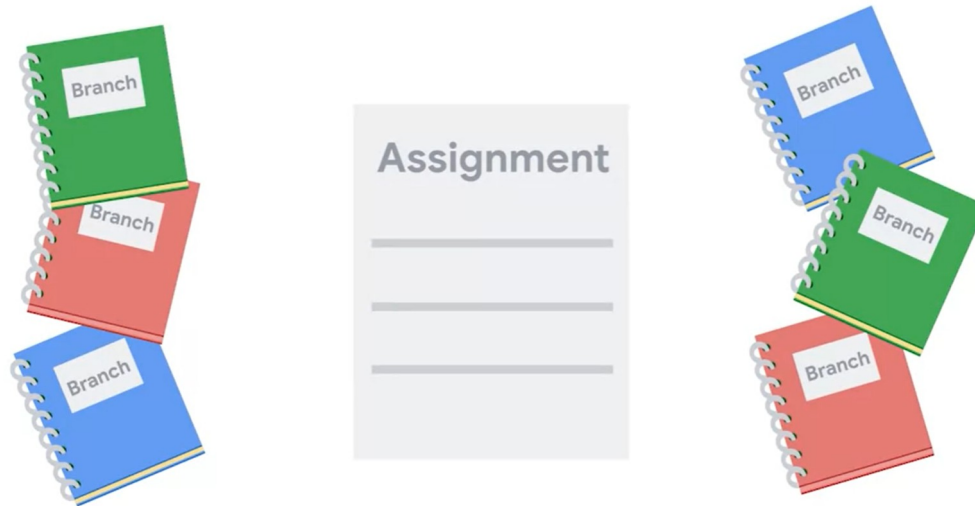


- Criar um repositório local
- Criar um arquivo no repositório local
- Adicionar o arquivo ao registro (transformá-lo em tracked)
- Realizar um commit
- Realizar uma alteração no arquivo e dar um commit direto, sem passar pela staging area (index)
- Realizar uma alteração no arquivo e levar o arquivo ao index
- Desfazer a alteração no arquivo

Branching and merging



- Basicamente, um **branch** é um ponteiro para um commit particular
- Representa uma linha de desenvolvimento independente num projeto
- O **branch** padrão criado pelo Git é chamado **master**
- É usado para experimentar novas ideias ou estratégias



```
# Verificando os branches existentes
(master)$ git branch
* master

# Criando um novo branch
(master)$ git branch nova-funcionalidade
(master)$ git branch
* master
  nova-funcionalidade

# Alternando para o novo branch
(master)$ git checkout nova-funcionalidade
Switched to branch 'nova-funcionalidade'

# Criando um novo branch e alternando para ele em um único comando
(nova-funcionalidade)$ git checkout -b funcionalidade-ainda-melhor
Switched to a new branch 'funcionalidade-ainda-melhor'

# Excluindo um branch sem alterações
(funcionalidade-ainda-melhor)$ git branch -d nova-funcionalidade
Deleted branch nova-funcionalidade (was 9e742f0).

# Realizando alguma alteração no programa
(funcionalidade-ainda-melhor)$ echo "#! /bin/bash" > script_genial.sh

# Enviando a alteração à staging area
(funcionalidade-ainda-melhor)$ git add script_genial.sh

# Realizando o commit
(funcionalidade-ainda-melhor)$ git commit -m 'Adicionada nova funcionalidade'
[funcionalidade-ainda-melhor 741266f] Adicionada nova funcionalidade
 1 file changed, 1 insertion(+)
 create mode 100644 script_genial.sh

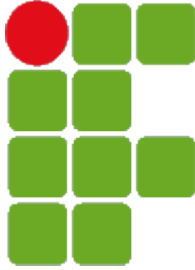
# Observando as entradas no log
(funcionalidade-ainda-melhor)$ git log -2
commit 741266f3009788402e615b91147e010a73f9e812 (HEAD -> funcionalidade-ainda-melhor)
Author: Diego Medeiros <diegomedeiros@ifsc.edu.br>
Date:   Fri Dec 1 00:59:32 2023 -0300

    Adicionada nova funcionalidade

commit 9e742f07dc4b26f48fda2553b1cfb7de8edba064 (master)
Author: Diego Medeiros <diegomedeiros@ifsc.edu.br>
Date:   Fri Dec 1 00:51:23 2023 -0300
```

Incluindo o `ls -l`

Trabalhando com branches



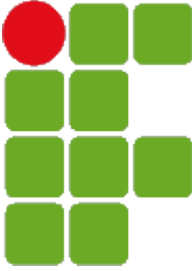
- Quando realizamos alterações em determinado branch, ela não será efetivada em outros
- Quando alternamos entre branches, todos os arquivos são atualizados (modificados, criados, deletados) para refletirem o estado daquele branch

```
# Verifica os arquivos no branch atual
(funcionalidade-ainda-melhor)$ ls -l
total 2
-rwxr-xr-x 1 Diego 197121 14 Dec  1 00:51 script.sh*
-rwxr-xr-x 1 Diego 197121 14 Dec  1 01:07 script_genial.sh*

# Alterna para a branch master
(funcionalidade-ainda-melhor)$ git checkout master
Switched to branch 'master'

# Verifica os arquivos no branch master
(master)$ ls -l
total 1
-rwxr-xr-x 1 Diego 197121 14 Dec  1 00:51 script.sh*
```


Merging branches



- Fluxo de trabalho típico
 - Cria um **branch** separado
 - Desenvolve uma nova funcionalidade
 - Realiza um **merge** para registrar as alterações

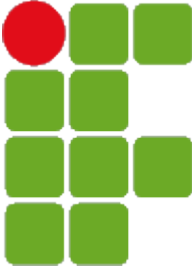
```
# Dado que está no branch master (perceba o asterisco *)
(master)$ git branch
funcionalidade-ainda-melhor
* master

# Realiza o merge com o branch funcionalidade-ainda-melhor
(master)$ git merge funcionalidade-ainda-melhor
Updating 9e742f0..741266f
Fast-forward
 script_genial.sh | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 script_genial.sh

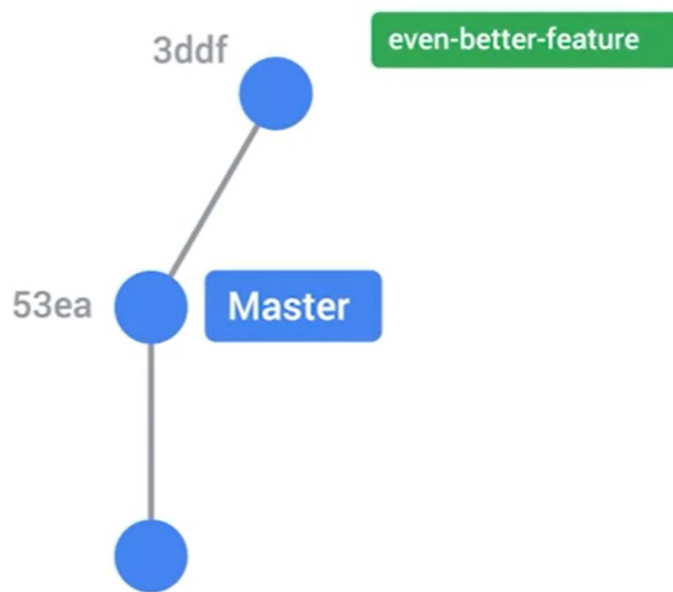
# Verifica o histórico de alterações
(master)$ git log -1
commit 741266f3009788402e615b91147e010a73f9e812 (HEAD -> master, funcionalidade-ainda-melhor)
Author: Diego Medeiros <diegomedeiros@ifsc.edu.br>
Date:   Fri Dec 1 00:59:32 2023 -0300
```

Adicionada nova funcionalidade

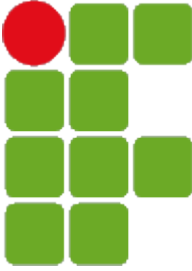
Algoritmos para merge



- Duas possibilidades:
 - Fast-forward merge: alteração feita no branch é diretamente atualizada
 - Three-way merge: alteração feita precisa ser confrontada com outras



Merge conflict



- Caso a mesma linha seja alterada em dois branches, o merge não é aplicado
- Ao abrir o editor de textos, uma interface interativa permite escolher qual versão usar

```
# Verifica o arquivo no branch master
(master)$ cat script_genial.sh
#!/bin/bash

# Faz uma alteração no código
(master) $ echo "ls -l" >> script_genial.sh

# Salva a alteração realizada
(master)$ git commit -a -m 'Adicionado o ls -l'
[master 2a565e5] Adicionado o ls -l
1 file changed, 1 insertion(+)

# Indo para o branch alternativo
(master)$ git checkout funcionalidade-ainda-melhor
Switched to branch 'funcionalidade-ainda-melhor'

# Verifica a script no branch alternativo - Como esperado script não foi alterada
(funcionalidade-ainda-melhor)$ cat script_genial.sh
#!/bin/bash

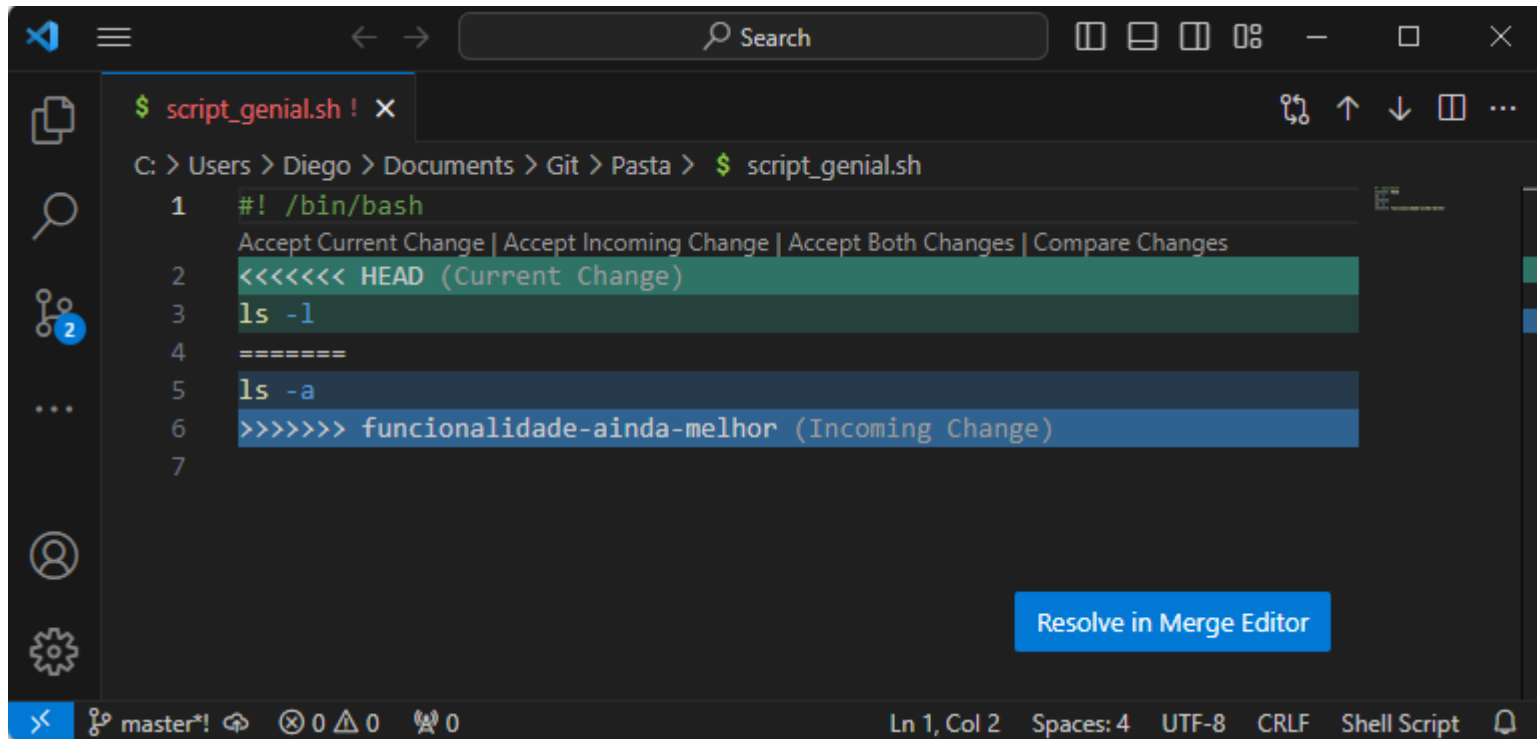
# Altera a script
(funcionalidade-ainda-melhor)$ echo "ls -a" >> script_genial.sh

# Salva a alteração no branch
(funcionalidade-ainda-melhor)$ git commit -a -m 'Adicionado o ls -a'
[funcionalidade-ainda-melhor 4e50ce1] Adicionado o ls -a
1 file changed, 1 insertion(+)

# Retorna para a branch master e tenta fazer um merge
(funcionalidade-ainda-melhor)$ git checkout master
Switched to branch 'master'

(master)$ git merge funcionalidade-ainda-melhor
Auto-merging script_genial.sh
CONFLICT (content): Merge conflict in script_genial.sh
Automatic merge failed; fix conflicts and then commit the result.
```

```
# Abre o arquivo que gerou o conflito
(master|MERGING)$ Code script_genial.sh
```

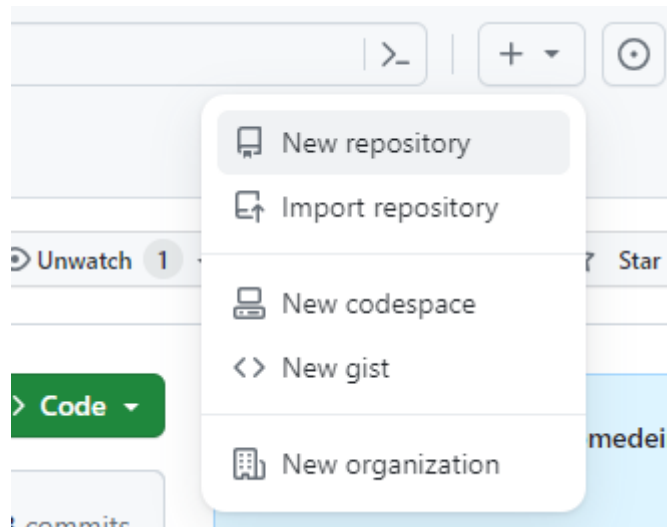


```
# Agora que o conflito foi solucionado, envia o arquivo à staging area
(master|MERGING)$ git add script_genial.sh
```

```
# Realiza o commit
(master|MERGING)$ git commit -m 'Resolvido o problema no merge'
[master 95bf8cd] Resolvido o problema no merge
```

```
# Verifica o resultado graficamente
(master)$ git log --graph --oneline
* 95bf8cd (HEAD -> master) Resolvido o problema no merge
| \
| * 4e50ce1 (funcionalidade-ainda-melhor) Adicionado o ls -a
| * | 2a565e5 Adicionado o ls -l
| /
* 741266f Adicionada nova funcionalidade
```

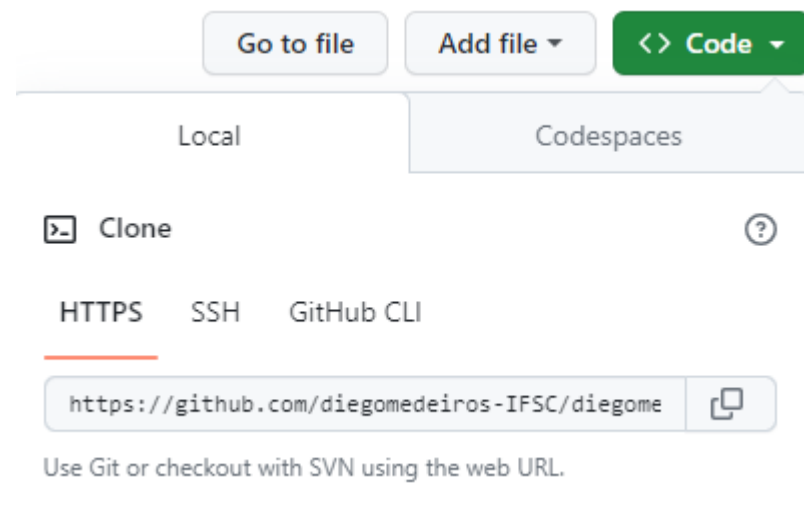
- Plataforma para hospedar seu repositório e criar um ambiente de colaboração entre desenvolvedores
- Usa o Git como VCS
- Entrar em github.com, logar e criar um novo repositório



- Incluir o nome do repositório, uma descrição, marcar a opção “Add a README file”, e clicar em “Create repository”



- Ir no botão verde “<> Code”, e copiar a URL disponível em HTTPS



```
# Traz ao diretório atual uma cópia do projeto
$ git clone https://github.com/diegomedeiros-IFSC/diegomedeiros-IFSC.git
Cloning into 'diegomedeiros-IFSC'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
Resolving deltas: 100% (1/1), done.


# Entra no diretório criado
$ cd diegomedeiros-IFSC/
(main)$ ls -l
total 4
-rw-r--r-- 1 Diego 197121 905 Dec  1 01:49 README.md

# Realiza uma alteração no arquivo, e salva o snapshot no branch main do repositório local
(main)$ echo "Este é o arquivo README.md" >> README.md
(main)$ git commit -a -m 'Adicionando uma descrição ao final do arquivo'
[main 560fa97] Adicionando um ponto ao final do arquivo
 1 file changed, 1 insertion(+)

# Envia o snapshot atual para o servidor
(main)$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 296 bytes | 296.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/diegomedeiros-IFSC/diegomedeiros-IFSC.git
afb090a..560fa97  main -> main
```

Problema de autenticação no git push

1

 **diegomedeiros-IFSC** ×

😊 Set status

👤 Your profile

➕ Add account

📁 Your repositories

📁 Your projects

📁 Your organizations

🌐 Your enterprises

⭐ Your stars

💖 Your sponsors

🔗 Your gists

📈 Upgrade

🌐 Try Enterprise

🤖 Copilot

🔗 Feature preview

⚙ Settings

📖 GitHub Docs

🔗 GitHub Support

🚪 Sign out

2

✉ Emails

🔒 Password and authentication

🔌 Sessions

🔑 SSH and GPG keys

🏢 Organizations

🌐 Enterprises

⚠ Moderation

Code, planning, and automation

📁 Repositories

📁 Codespaces

📦 Packages

🤖 Copilot

📄 Pages

↩ Saved replies

Security

🔒 Code security and analysis

Integrations

🔌 Applications

🕒 Scheduled reminders

Archives

📖 Security log

📖 Sponsorship log

<> Developer settings

3

🔌 GitHub Apps

👤 OAuth Apps

🔑 **Personal access tokens**

Fine-grained tokens Beta

Tokens (classic)

4

Generate new token ▼

Generate new token Beta

Fine-grained, repo-scoped

Generate new token (classic)

For general use

5

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used to [authenticate to the API over Basic Authentication](#) over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#) over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#) over HTTPS.

Note

Inserir anotação

What's this token for?

Expiration *

30 days ⌵ The token will expire on Thu, Jan 11 2024

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#)

☒ repo

Full control of private repositories

☒ repo:status

Access commit status

☒ repo_deployment

Access deployment status

☒ public_repo

Access public repositories

☒ repo:invite

Access repository invitations

☒ security_events

Read and write security events

Usar o token gerado no lugar do password no git push

31

Usando um repositório remoto



- Time desenvolvendo um projeto conjunto
- Cada pessoa é responsável por uma parte do código
- Frequentemente, todos precisam enviar suas atualizações
- Alterações são combinadas para garantir que tudo é compatível
- Comando `git fetch` traz alterações do repo remoto mas não atualiza os arquivos
- Comando `git pull` traz alterações do repo remoto e atualiza os arquivos

```
# Mostra informações do repositório atual e do repositório remoto
(master)$ git remote show origin
* remote origin
  Fetch URL: https://github.com/diegomedeiros-IFSC/IC07862-2023-2.git
  Push URL: https://github.com/diegomedeiros-IFSC/IC07862-2023-2.git
  HEAD branch: main
  Remote branches:
    main tracked
  Local branch configured for 'git pull':
    main merges with remote main
  Local ref configured for 'git push':
    main pushes to main (up to date)

# Se algo foi alterado no repositório remoto, a desatualização é acusada
(master)$ git remote show origin
* remote origin
  Fetch URL: https://github.com/diegomedeiros-IFSC/IC07862-2023-2.git
  Push URL: https://github.com/diegomedeiros-IFSC/IC07862-2023-2.git
  HEAD branch: main
  Remote branches:
    main tracked
  Local branch configured for 'git pull':
    main merges with remote main
  Local ref configured for 'git push':
    main pushes to main (local out of date)
```


Usando um repositório remoto

Traz os commits realizados no repositório remoto sem fazer um merge, para que possamos entender as alterações

```
(main)$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 670 bytes | 10.00 KiB/s, done.
From https://github.com/diegomedeiros-IFSC/diegomedeiros-IFSC
 1bb2778..aa4f683  main    -> origin/main
```

Verifica a diferença entre os snapshots

```
(main)$ git log origin/main -2
commit aa4f683568a5b531c6c01ebf5798728181e1a295 (origin/main, origin/HEAD)
Author: diegomedeiros-IFSC <34062049+diegomedeiros-IFSC@users.noreply.github.com>
Date:   Fri Dec 1 02:15:00 2023 -0300
```

Update README.md

Mensagem de FIM

```
commit 1bb27780c288fe9ed26126293be14d28e78b89ad (HEAD -> main)
Author: diegomedeiros-IFSC <34062049+diegomedeiros-IFSC@users.noreply.github.com>
Date:   Fri Dec 1 02:10:46 2023 -0300
```

Update README.md

Retirei o .

Realizando o merge para a versão mais atual

```
(main)$ git merge origin/main
Updating 1bb2778..aa4f683
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
```

Verifica como está o log agora

```
(main)$ git log -1
commit aa4f683568a5b531c6c01ebf5798728181e1a295 (HEAD -> main, origin/main, origin/HEAD)
Author: diegomedeiros-IFSC <34062049+diegomedeiros-IFSC@users.noreply.github.com>
Date:   Fri Dec 1 02:15:00 2023 -0300
```

Update README.md

Mensagem de FIM

Resolvendo conflitos

- Quando alterações acontecem no repositório remoto e no local, o git push não consegue resolver os conflitos
- Temos que resolver o conflito para conseguir enviar as alterações para o repositório remoto

```
# Realizar alterações no arquivo README.md local e remotos, realizar o commit

# Tentando executar um push para enviar as alterações para o repositório remoto
(main)$ git push
To https://github.com/diegomedeiros-IFSC/IC07862-2023-2.git
! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/diegomedeiros-IFSC/IC07862-2023-2.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

# Podemos tentar usar o git pull para sincronizar o branch remoto local com o repositório remoto antes de push
(main)$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 763 bytes | 13.00 KiB/s, done.
From https://github.com/diegomedeiros-IFSC/IC07862-2023-2
   776df21..0bc65bc  main       -> origin/main
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

# Tentou dar um merge automático mas não conseguiu

# Podemos ver o log para tentar entender o problema
(main)$ git log --graph --oneline --all
* 2f72f72 (HEAD -> main) Adicionando outro comentário
| * 0bc65bc (origin/main, origin/HEAD) Update README.md
| /
* 776df21 Create script2.sh
* eac285d Create script.sh
```

Buscando detalhes das alterações (opção -p)

```
(main)$ git log -p origin/main -2
```

```
commit 0bc65bc29af7b5c7788a8ce6aea39f53bfa2bf5a (origin/main, origin/HEAD)
Author: diegomedeiros-IFSC <34062049+diegomedeiros-IFSC@users.noreply.github.com>
Date: Thu Nov 30 17:10:54 2023 -0300
```

Update README.md

```
diff --git a/README.md b/README.md
index a83ad9a..87fa6e4 100644
--- a/README.md
+++ b/README.md
@@ -1,2 +1,2 @@
 # IC07862-2023-2
 -Repositório da turma de 2023-2 de ICO
```

```
commit 0bc65bc29af7b5c7788a8ce6aea39f53bfa2bf5a (origin/main, origin/HEAD)
Author: diegomedeiros-IFSC <34062049+diegomedeiros-IFSC@users.noreply.github.com>
Date: Thu Nov 30 17:10:54 2023 -0300
```

Update README.md

```
diff --git a/README.md b/README.md
index a83ad9a..87fa6e4 100644
--- a/README.md
+++ b/README.md
@@ -1,2 +1,2 @@
 # IC07862-2023-2
```

```
-Repositório da turma de 2023-2 de ICO
+Repositório da turma de 2023-2 de ICO - Adicionando um comentário
```

Aqui está o conflito

Executamos um merge do repositório remoto para o local

```
(main)$ git merge origin/main
```

Mesclagem automática de README.md

CONFLITO (conteúdo): conflito de mesclagem em README.md

Automatic merge failed; fix conflicts and then commit the result.

Abrimos o arquivo README.md com o editor de textos e corrigimos o conflito

```
(main)$ Code README.md
```

Registramos a alteração

```
(main)$ git add README.md
```

```
(main)$ git commit
```

```
[main 36d7f87] Merge branch 'main' of https://github.com/diegomedeiros-IFSC/IC07862-2023-2
```

E finalmente podemos enviar as alterações para o repositório remoto

```
(main)$ git push
```

Enumerating objects: 8, done.

Counting objects: 100% (8/8), done.

Delta compression using up to 8 threads

Compressing objects: 100% (4/4), done.

Writing objects: 100% (4/4), 654 bytes | 654.00 KiB/s, done.

Total 4 (delta 0), reused 0 (delta 0), pack-reused 0

To https://github.com/diegomedeiros-IFSC/IC07862-2023-2.git

0bc65bc..36d7f87 main -> main

Olhando como está a situação agora

```
(main)$ git log --graph --oneline
```

```
* 36d7f87 (HEAD -> main, origin/main, origin/HEAD) Merge branch 'main' of https://github....
```

```
| \
| * 0bc65bc Update README.md
```

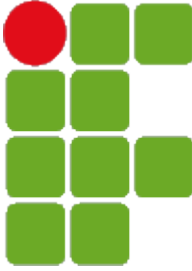
```
* | 2f72f72 Adicionando outro comentário
```

```
| /
```

```
* 776df21 Create script2.sh
```

```
* eac285d Create script.sh
```

Branches remotos



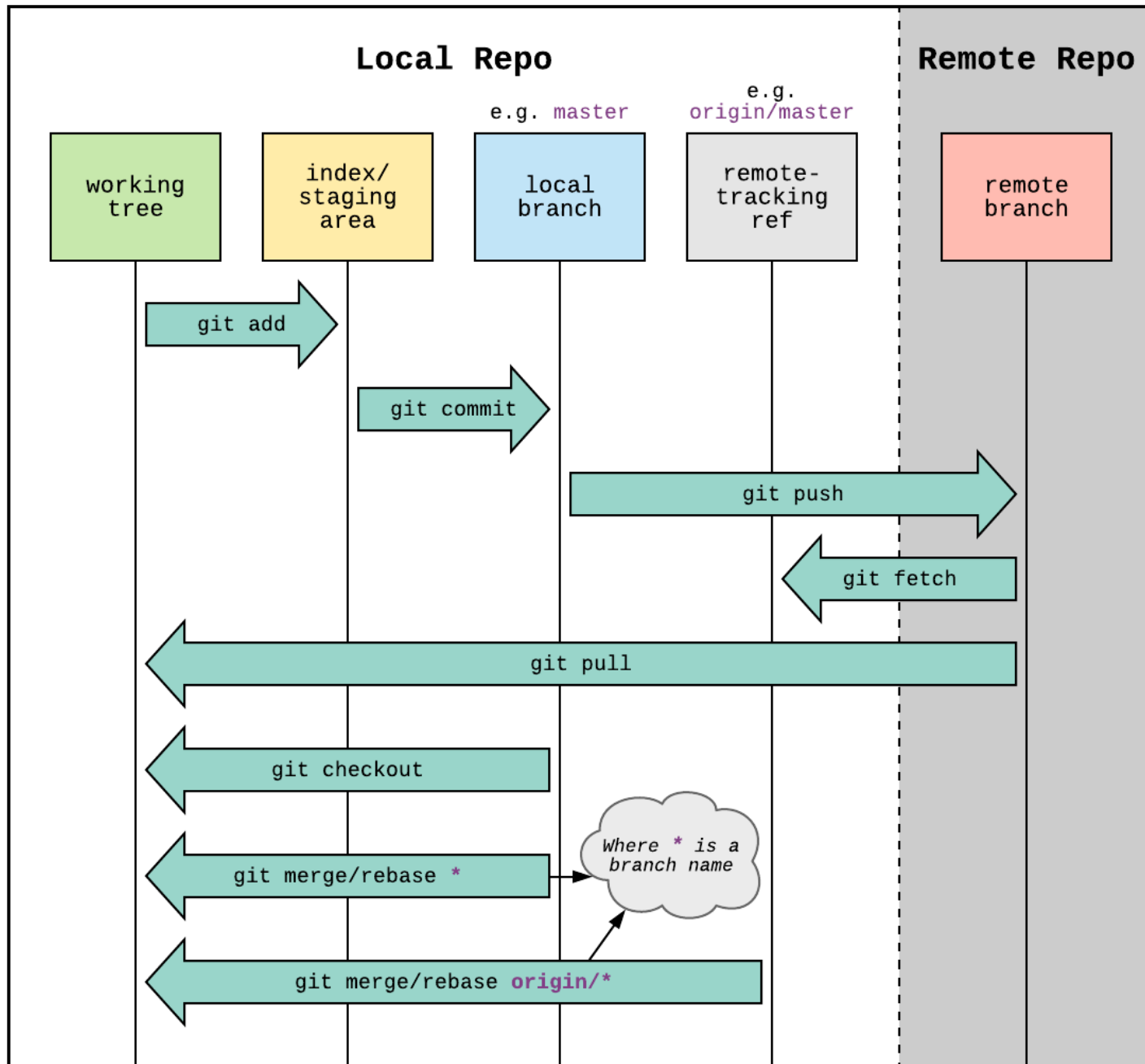
- Quando criamos branches locais para realizar alterações, precisamos de mais alguns comandos para salvar o novo branch no repositório remoto

```
# Criando um ramo local novo chamado "refactor"
(main)$ git checkout -b refactor
Switched to a new branch 'refactor'
```

```
# Alteramos a script e damos um commit
(main)$ echo "ls -a" » script.sh
(main)$ git commit -a -m 'Adicionando ls -a ao script.sh'
[refactor 3b146ca] Adicionando ls -a ao script.sh
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
# Como será a primeira vez que mandamos esse branch para o repositório remoto, precisamos indicar o novo branch
(main)$ git push -u origin refactor
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 338 bytes | 338.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'refactor' on GitHub by visiting:
remote:   https://github.com/diegomedeiros-IFSC/IC07862-2023-2/pull/new/refactor
remote:
To https://github.com/diegomedeiros-IFSC/IC07862-2023-2.git
 * [new branch]      refactor -> refactor
branch 'refactor' set up to track 'origin/refactor'.
```

Diagrama (local e remoto)



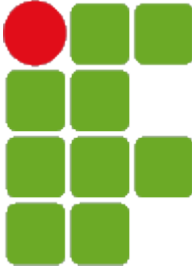
Fonte: [link](#)

Referências



- Inglês
 - **Coursera**: Curso base para a aula. Pago, mas conta com 7 dias grátis.
 - **Como criar seu Profile README**
 - **Udemy**: Grátis, 2h de aula
- Português
 - **EBAC**
 - **Cursa**: Grátis, 2h de aula
 - **Curso em vídeo**: Grátis, 20h de aula
 - **Git – guia prático**: ótimo guia rápido para consultas
 - **GIT – Sobre o REBASE**: Um ótimo guia sobre o comando REBASE

Atividade



- Fazer curso disponível em:
 - <https://github.com/WoMakersCode/git-e-github>