Curso de Extensão

Tecnologias Microsoft

UNICAMP

INF-0991

Controle de Versão

AULA 2

Prof. Dr. Rodrigo Bonacin

RBonacin@unicamp.br

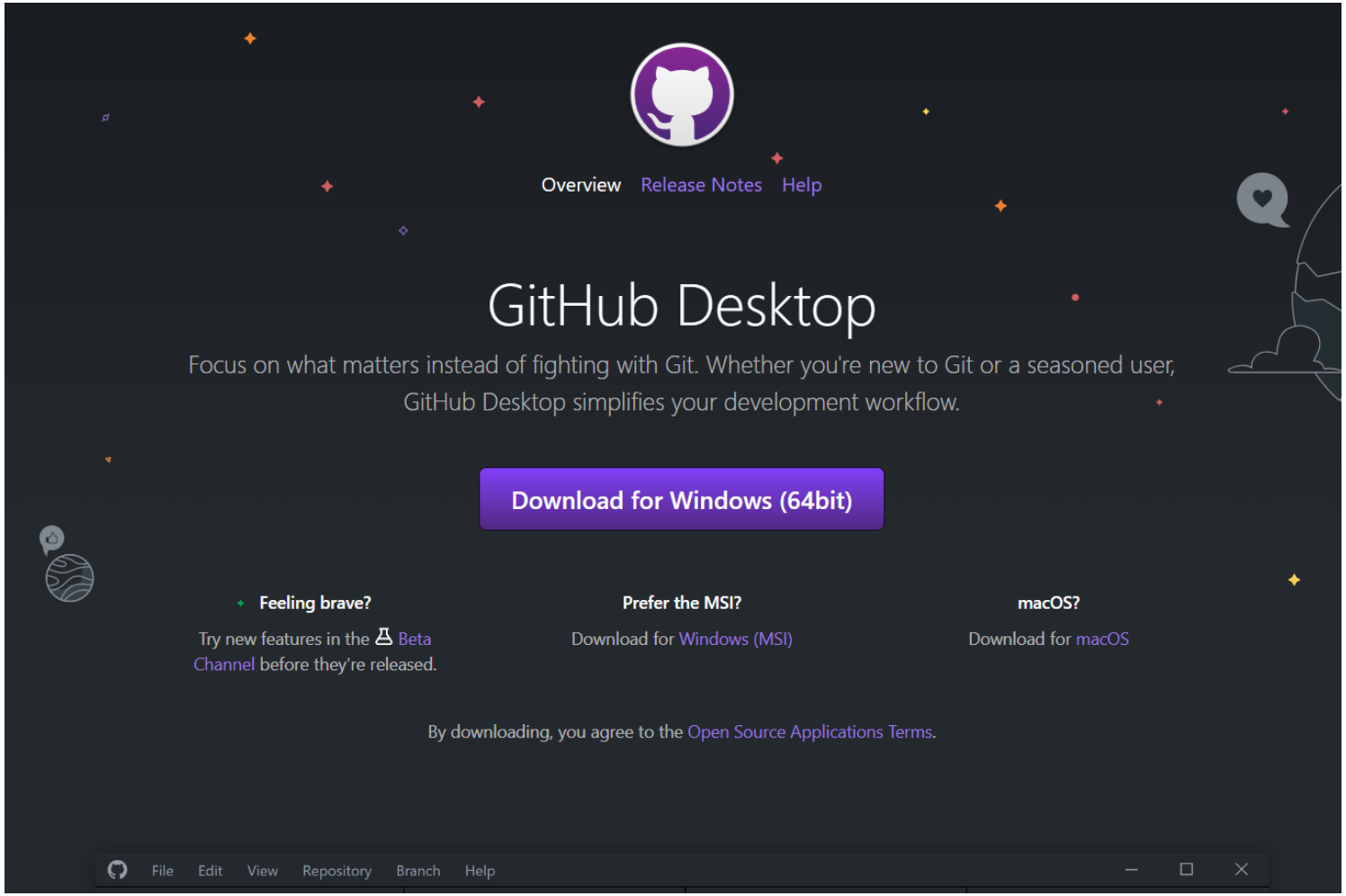
03 de Setembro de 2022

Agenda – Aula 2

1. Correção Atividade 1 (outro arquivo de slides)
2. Ferramentas
   - GitHub Desktop
   - Visual Studio Code com Git/GitHub
3. Boas práticas
4. Atividades Práticas com Ferramentas em Sala de Aula
5. Atividade 2

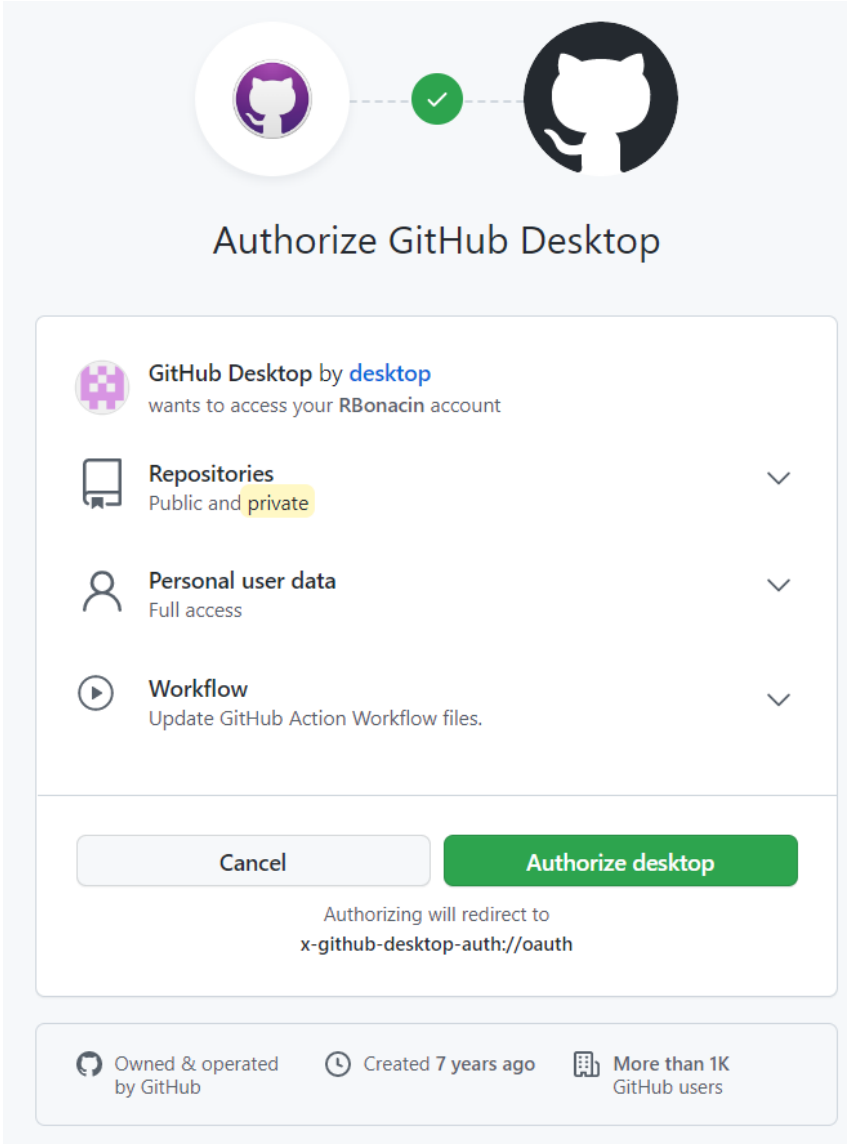
Ferramentas – GitHub Desktop

GitHub Desktop

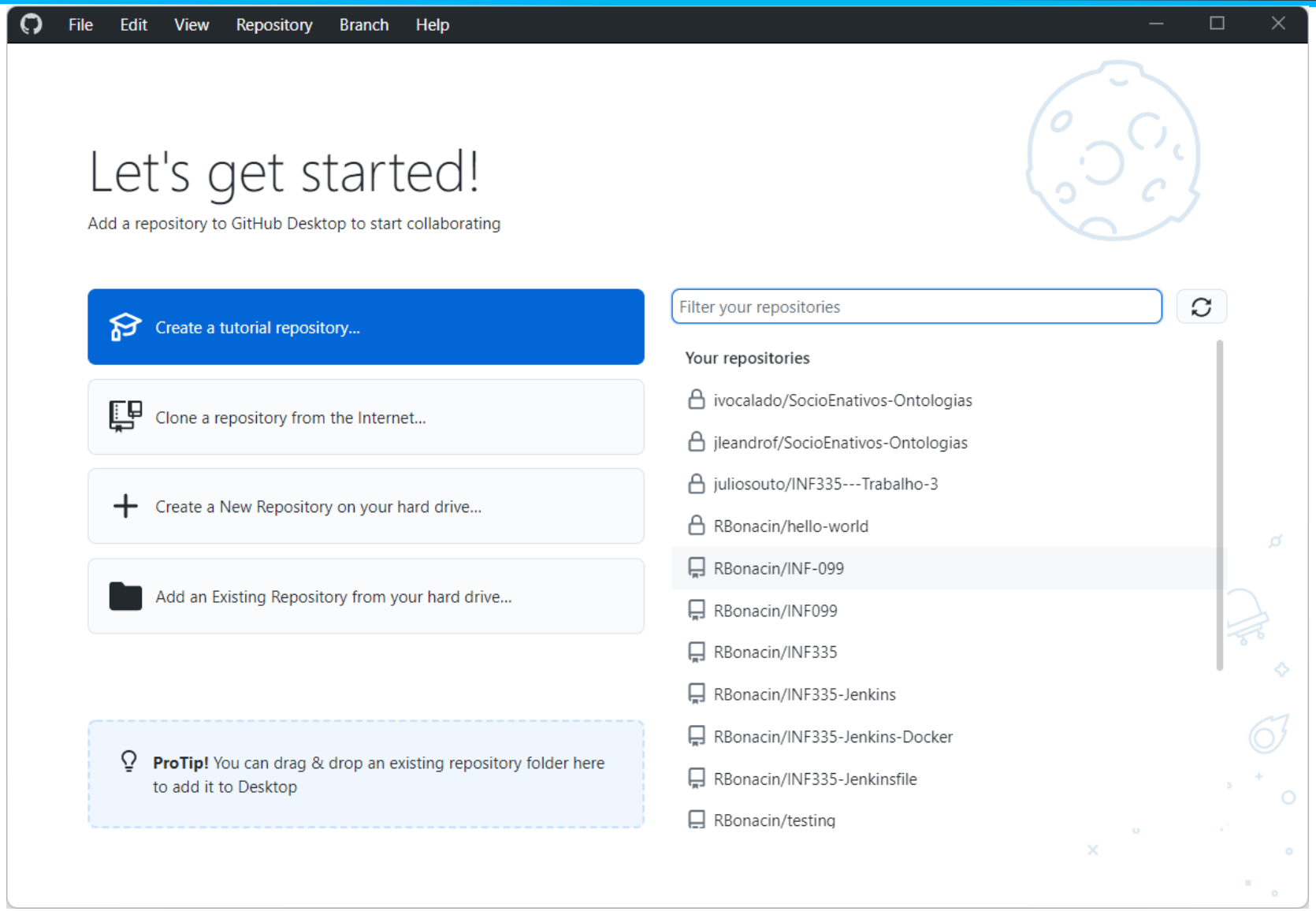


https://desktop.github.com/

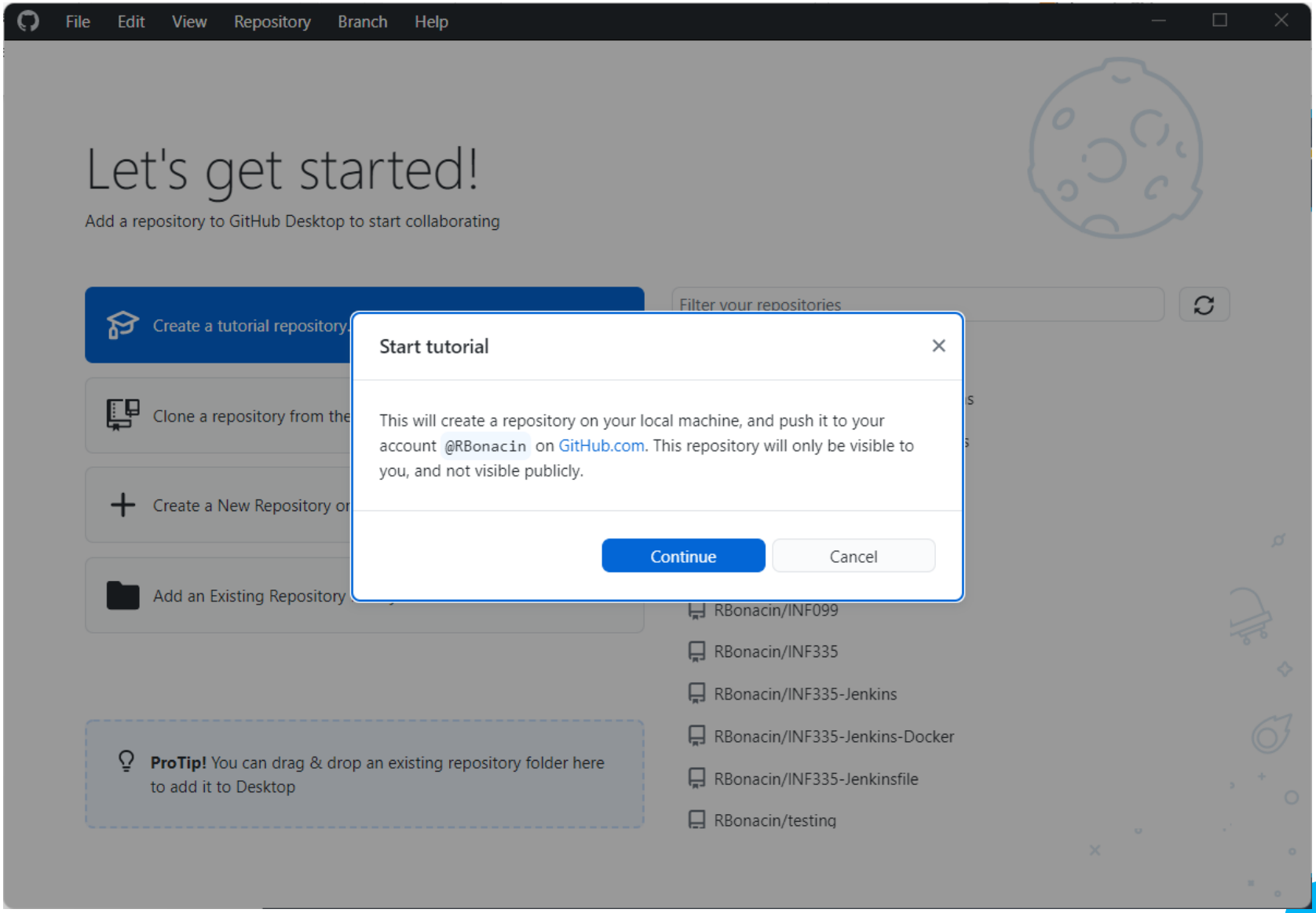
GitHub Desktop



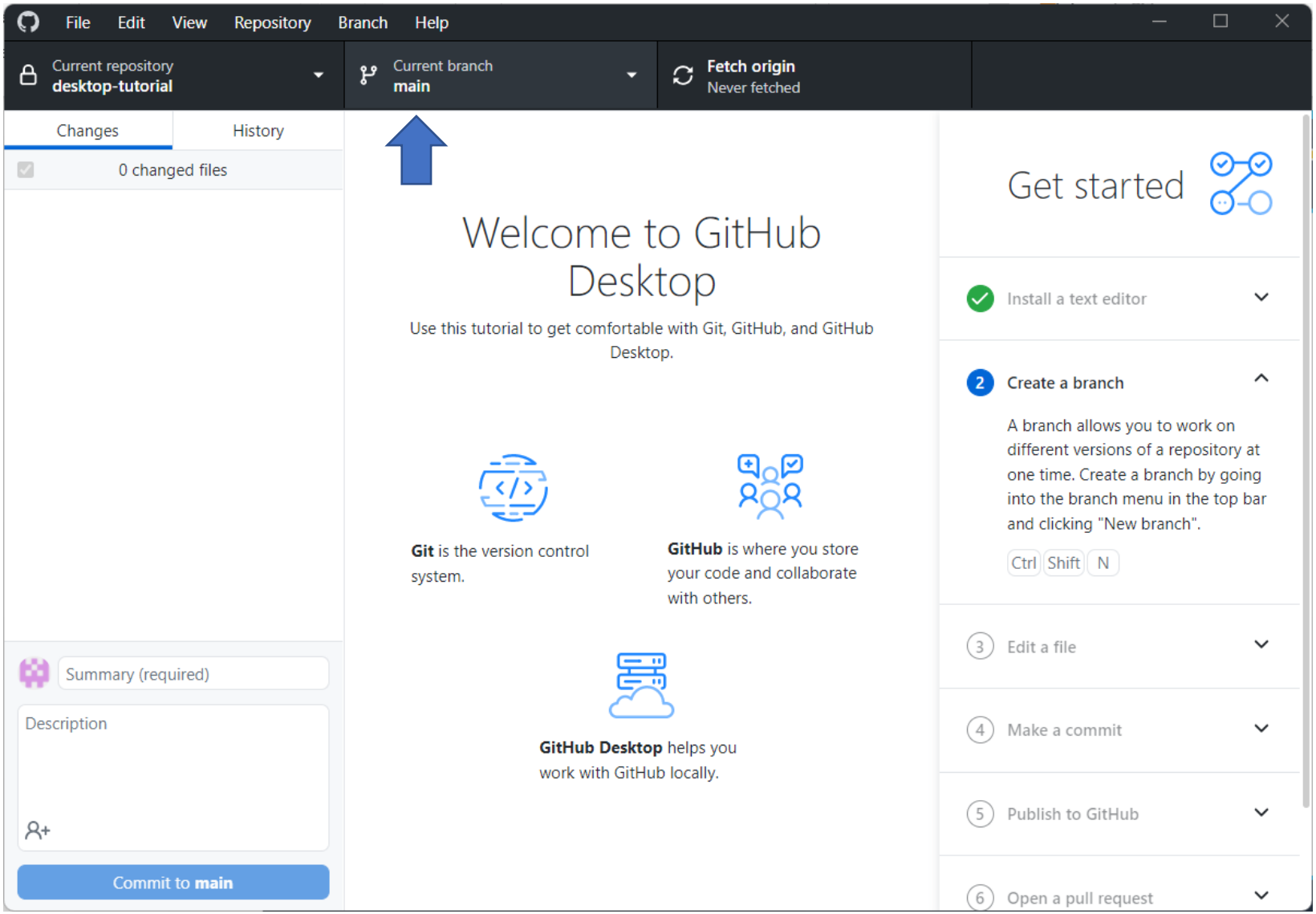
GitHub Desktop



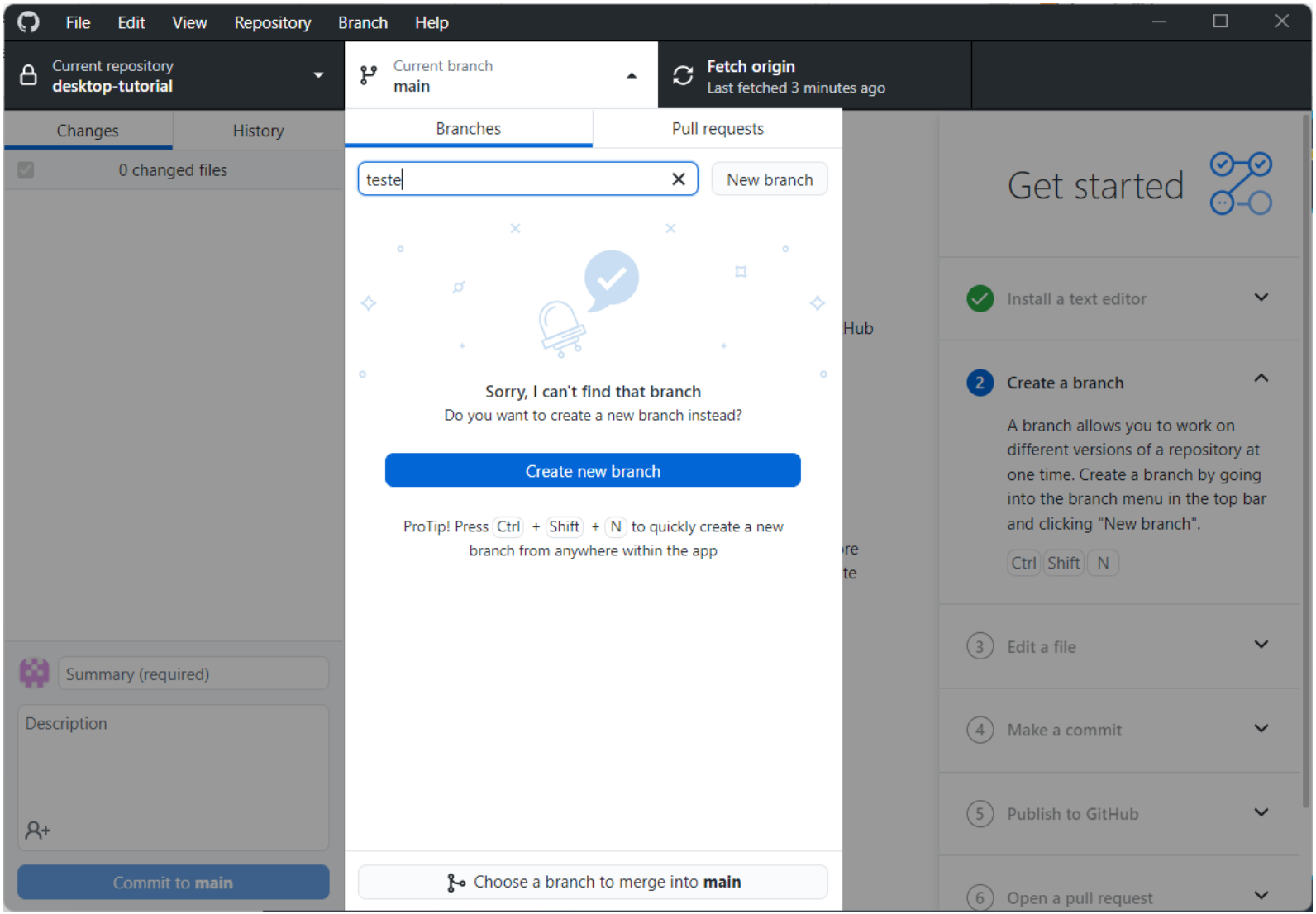
GitHub Desktop



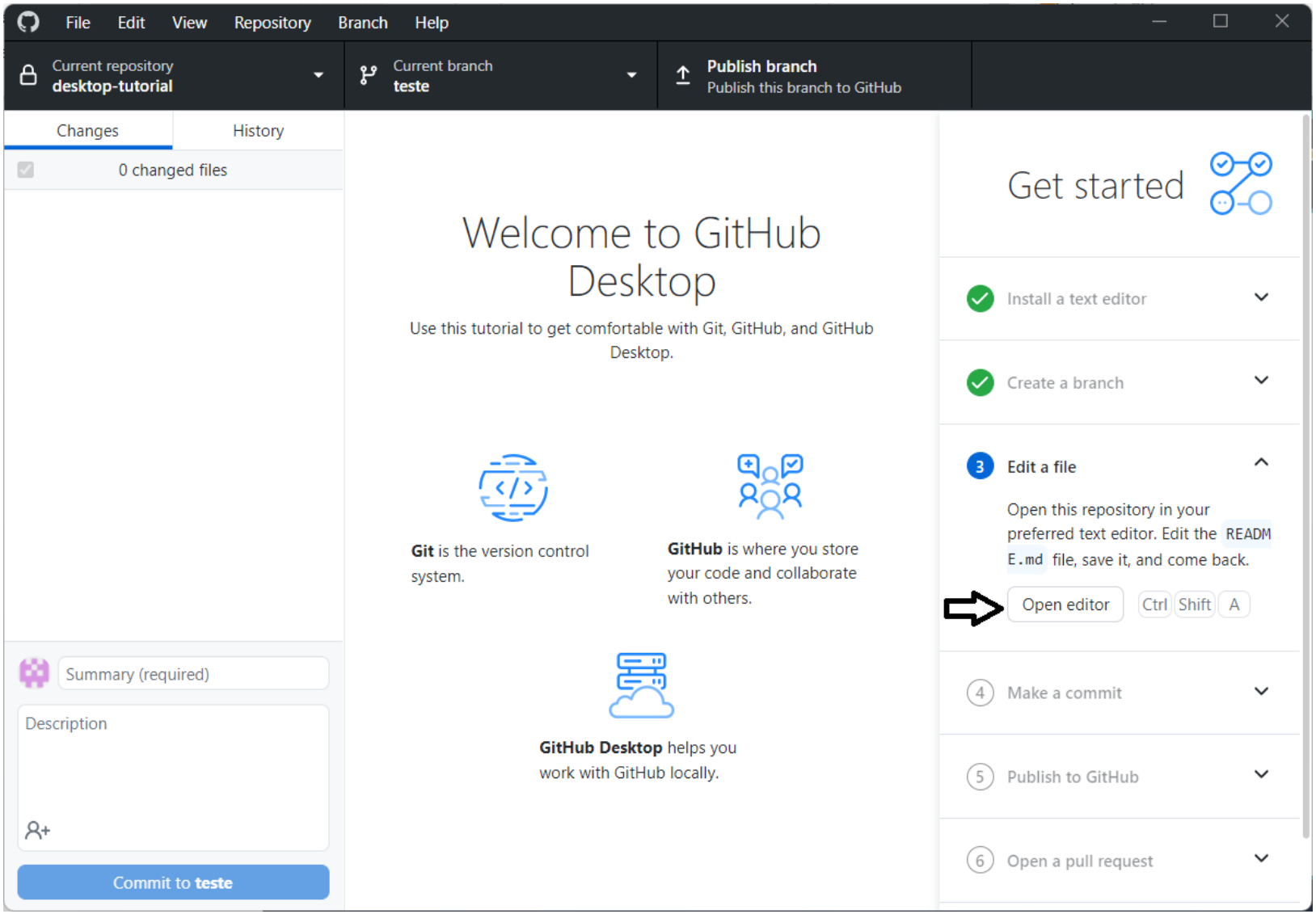
GitHub Desktop



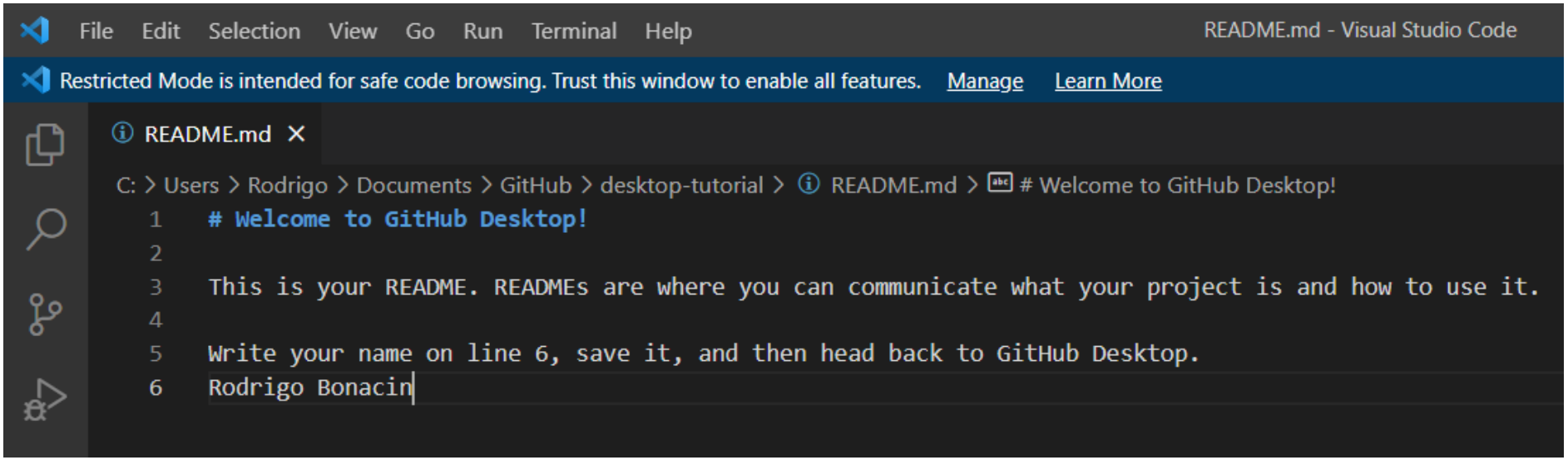
GitHub Desktop



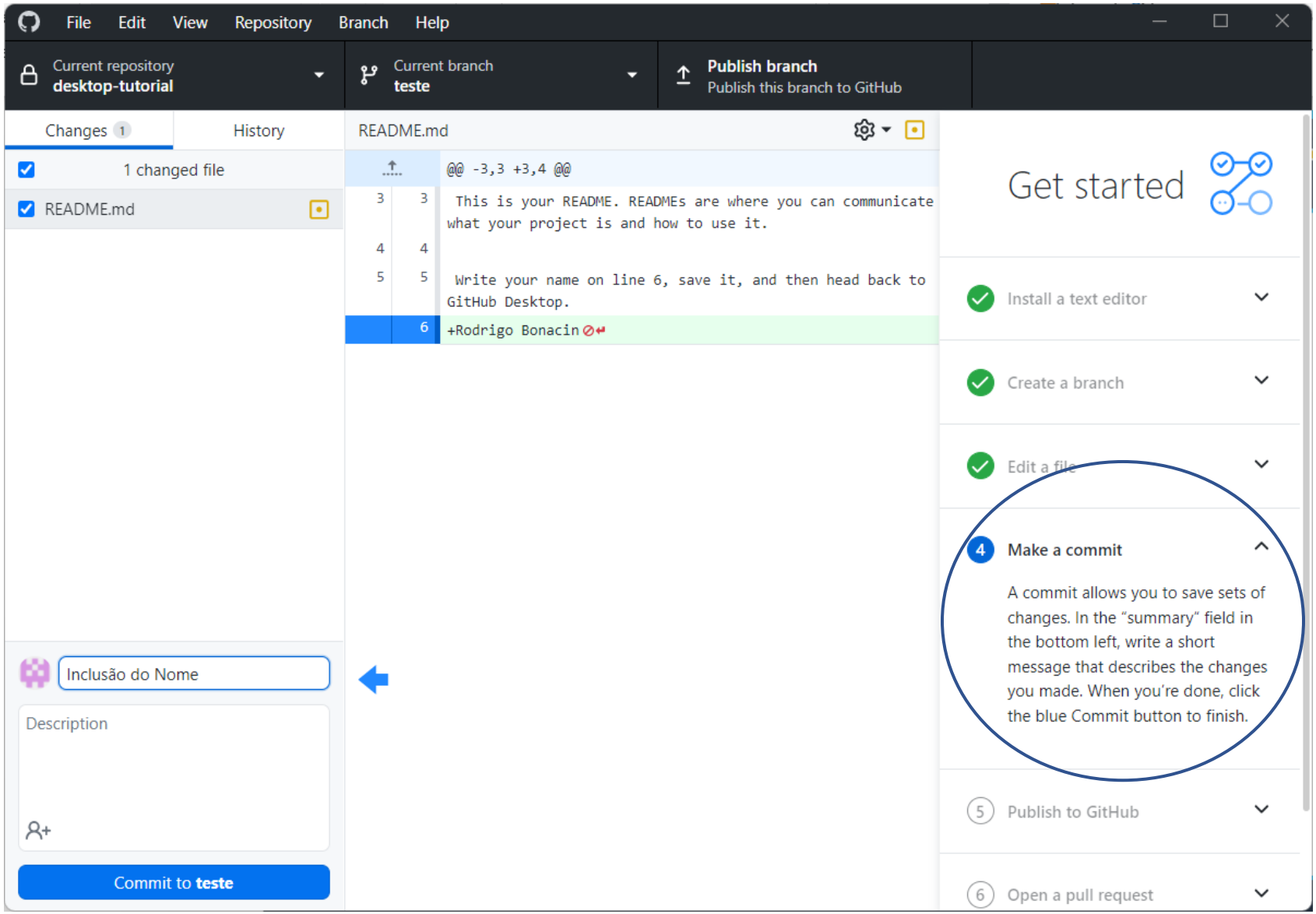
GitHub Desktop



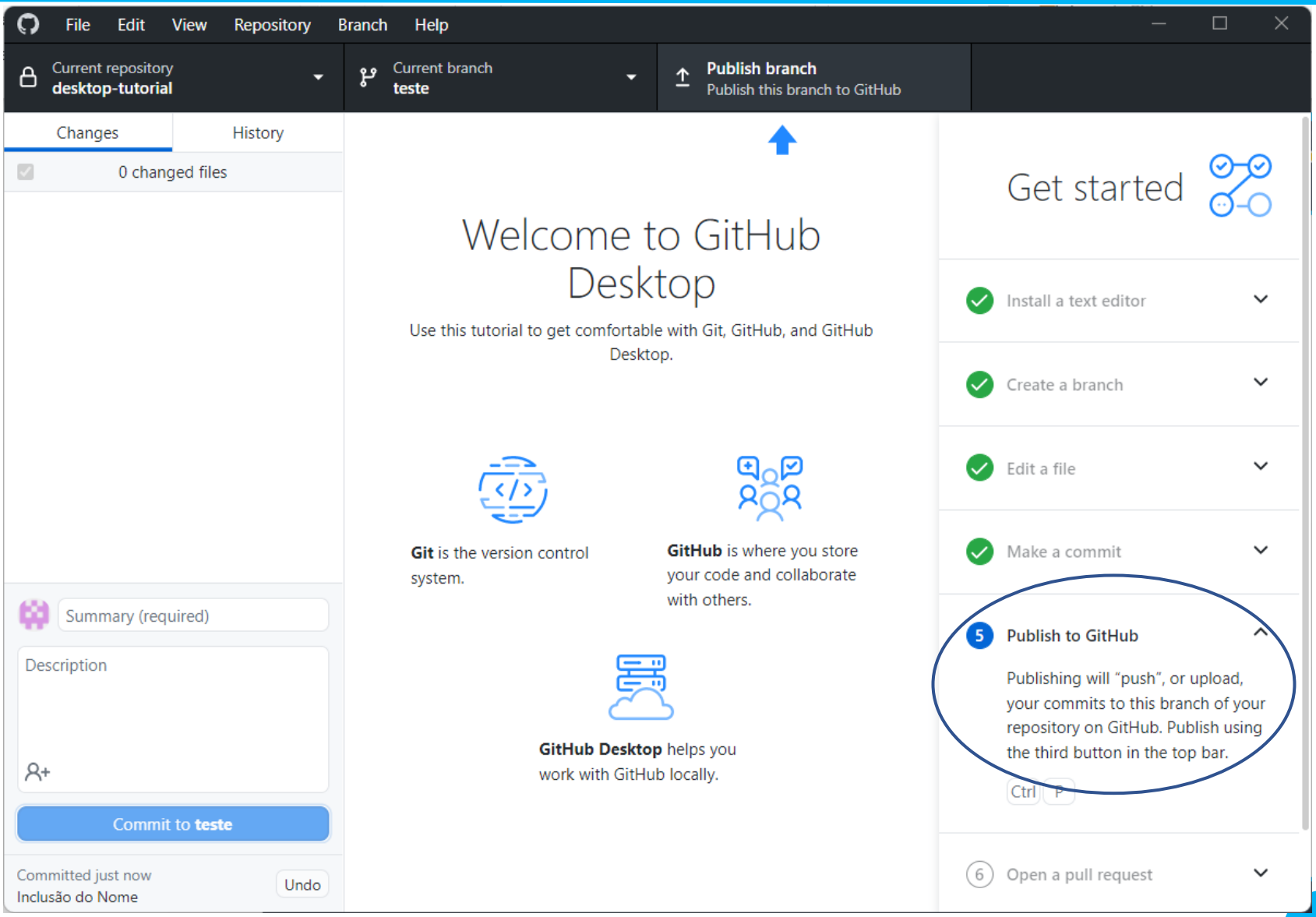
GitHub Desktop



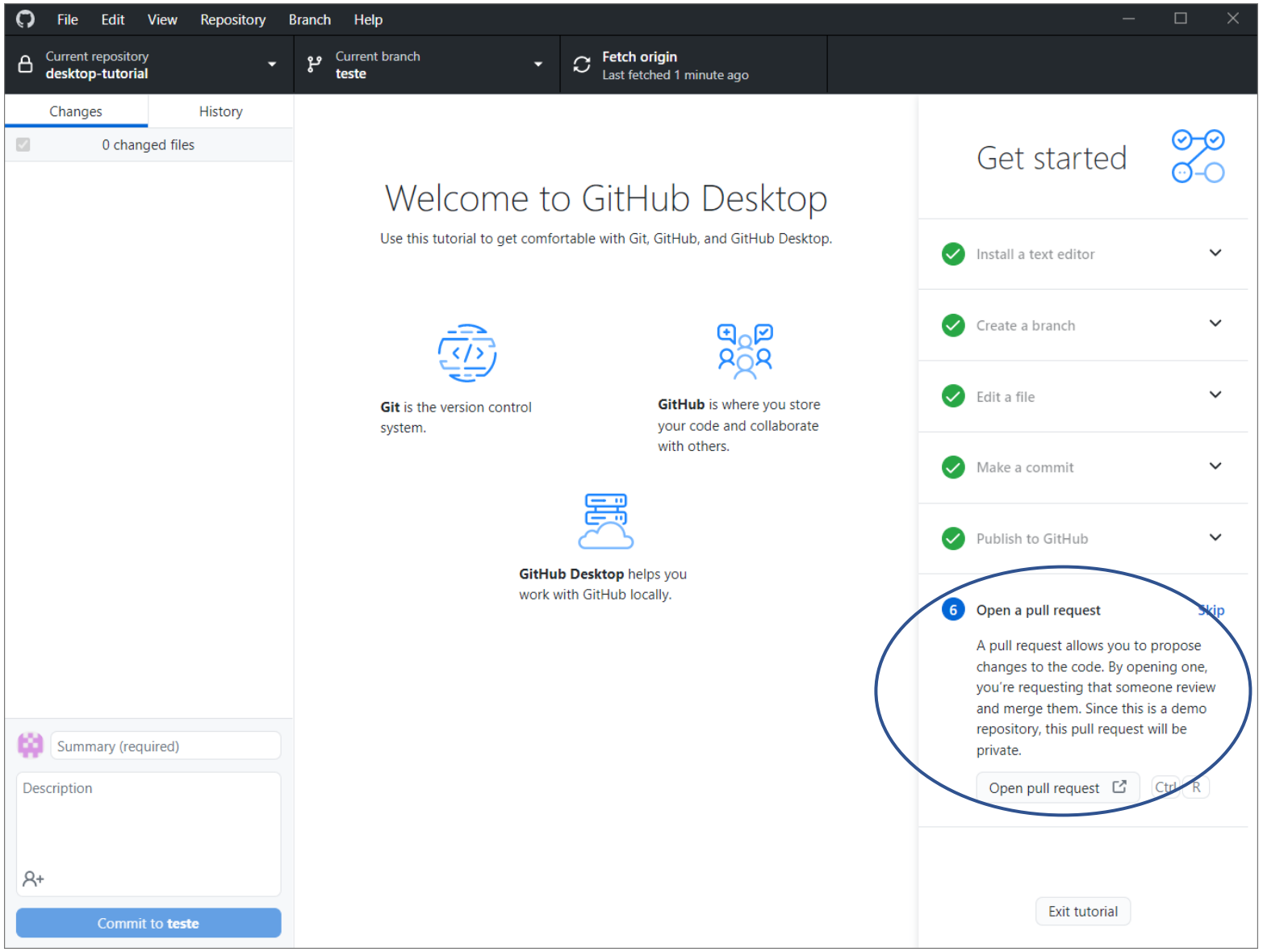
GitHub Desktop



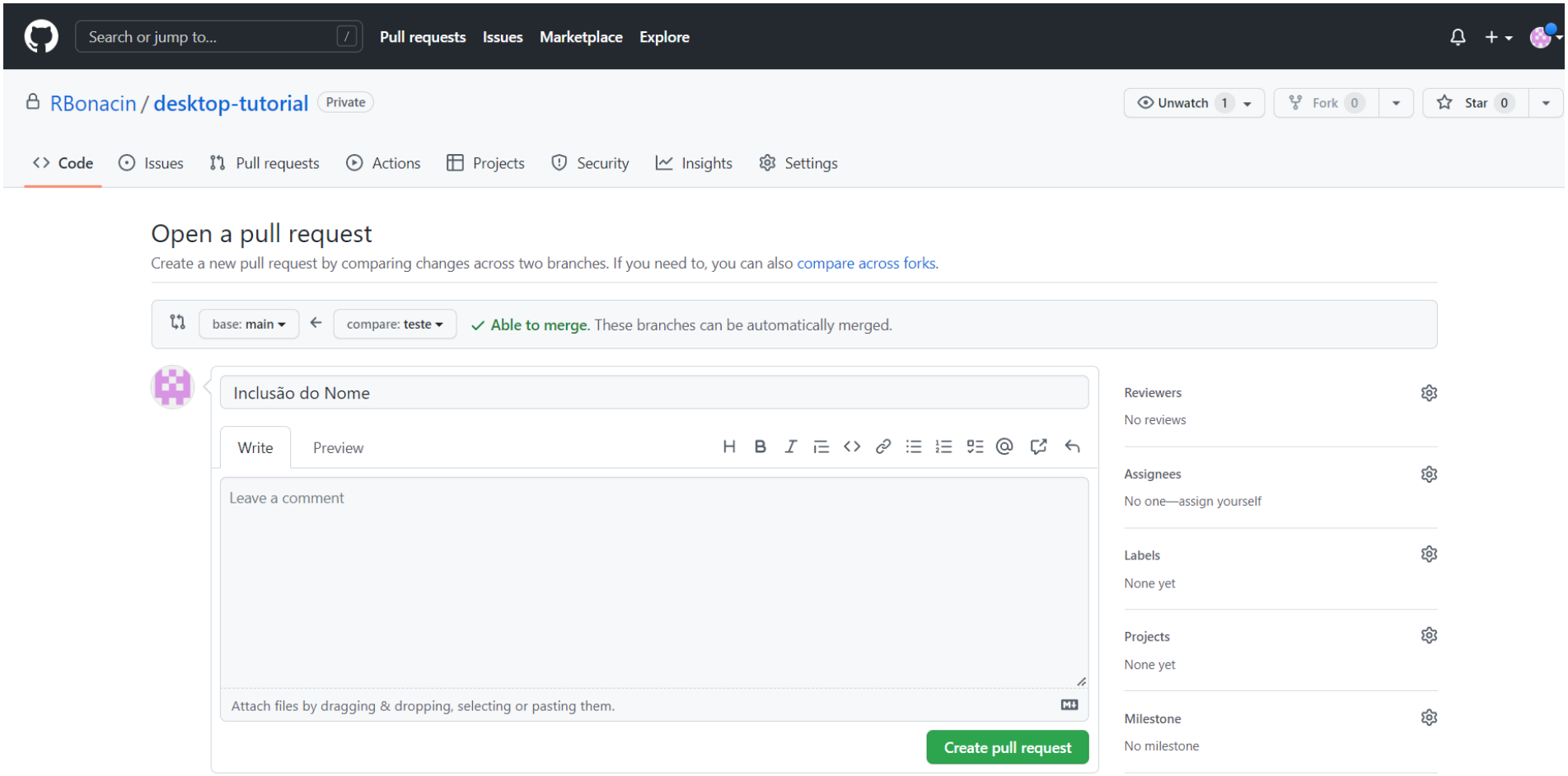
GitHub Desktop



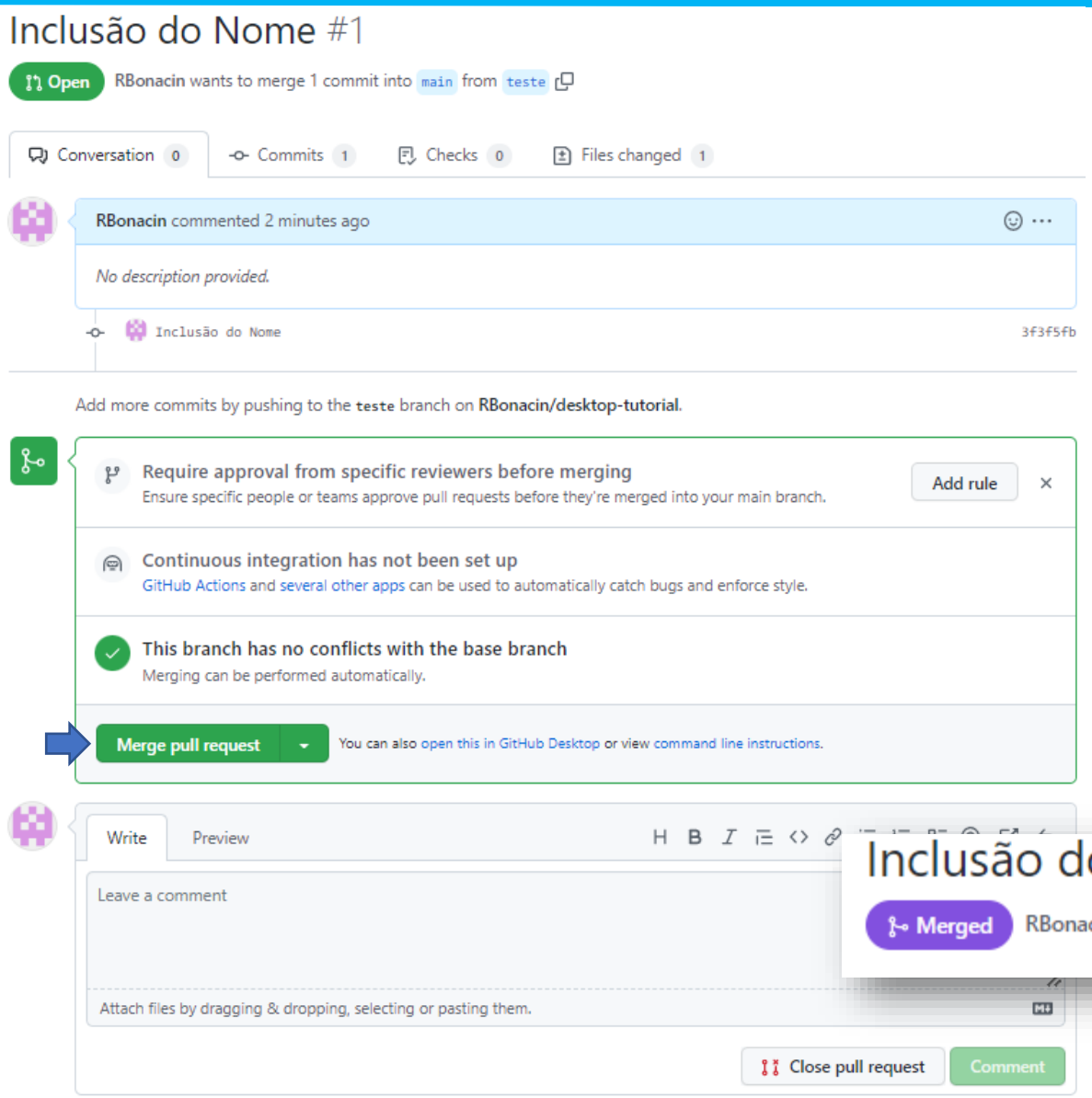
GitHub Desktop

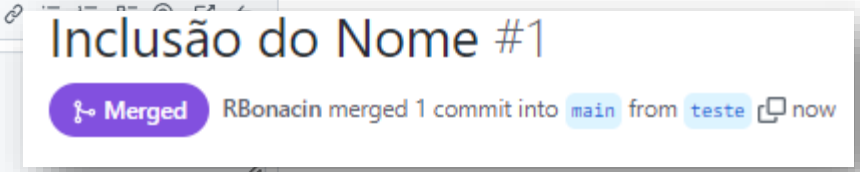


GitHub Desktop

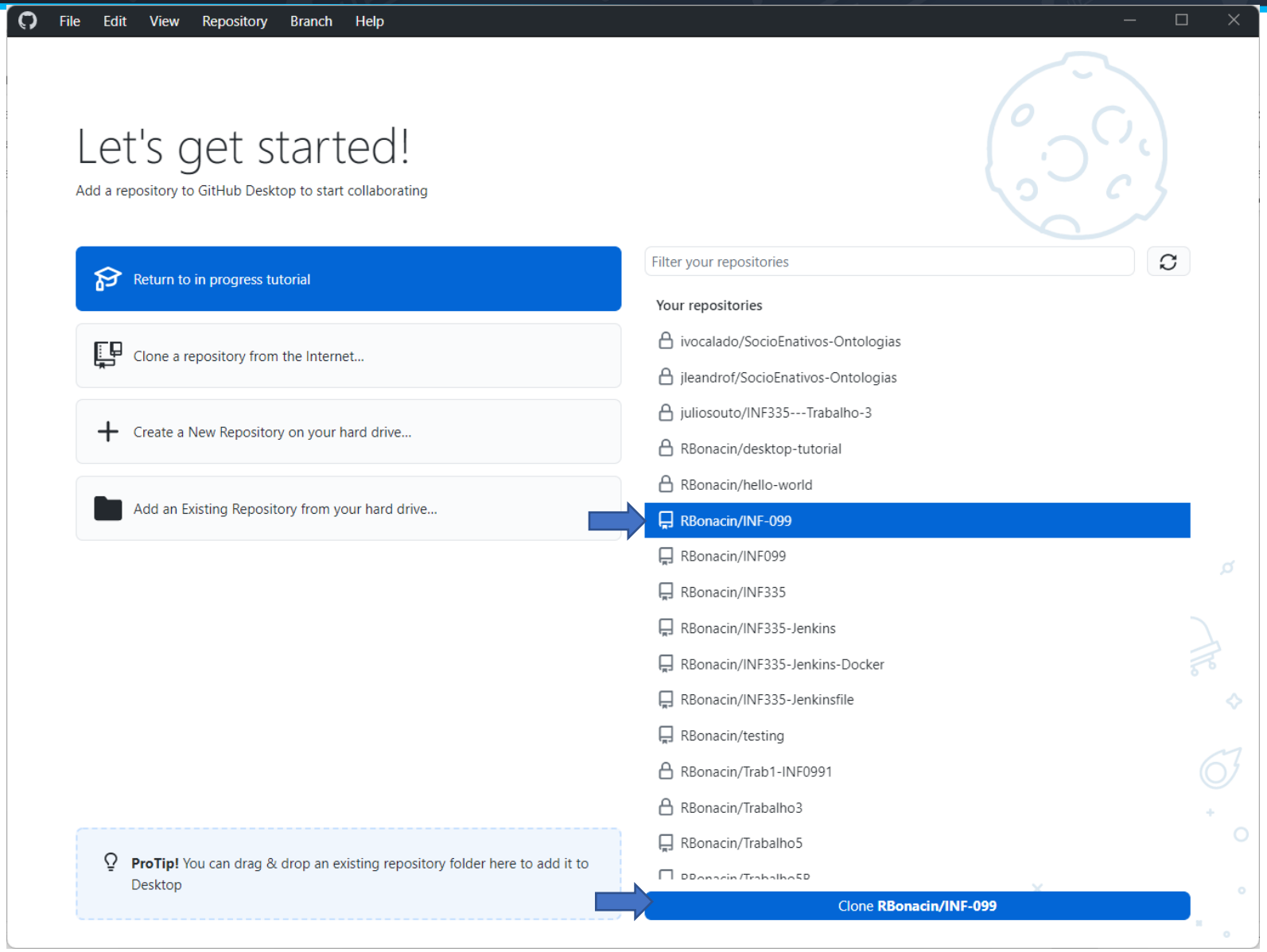


GitHub Desktop

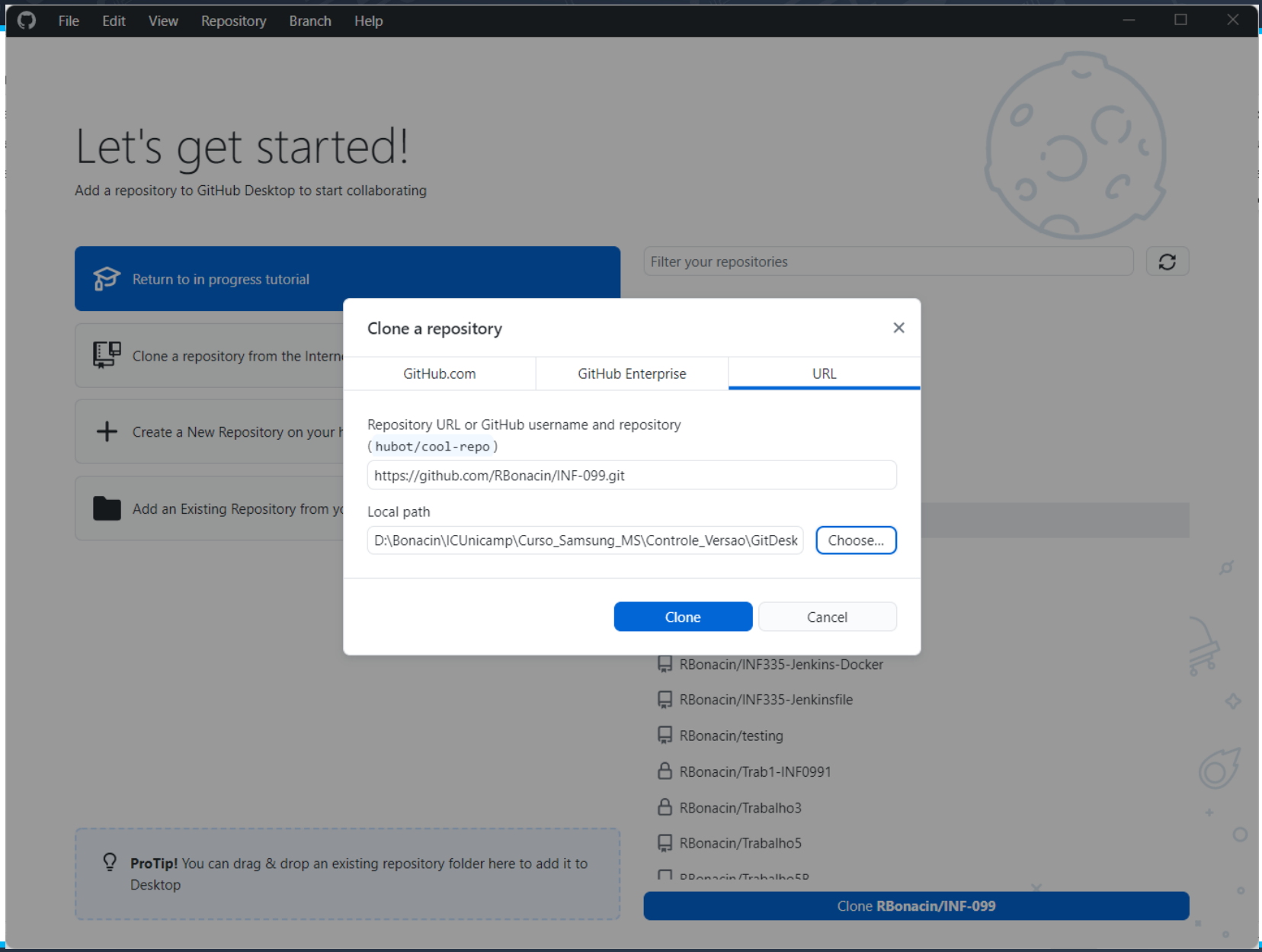




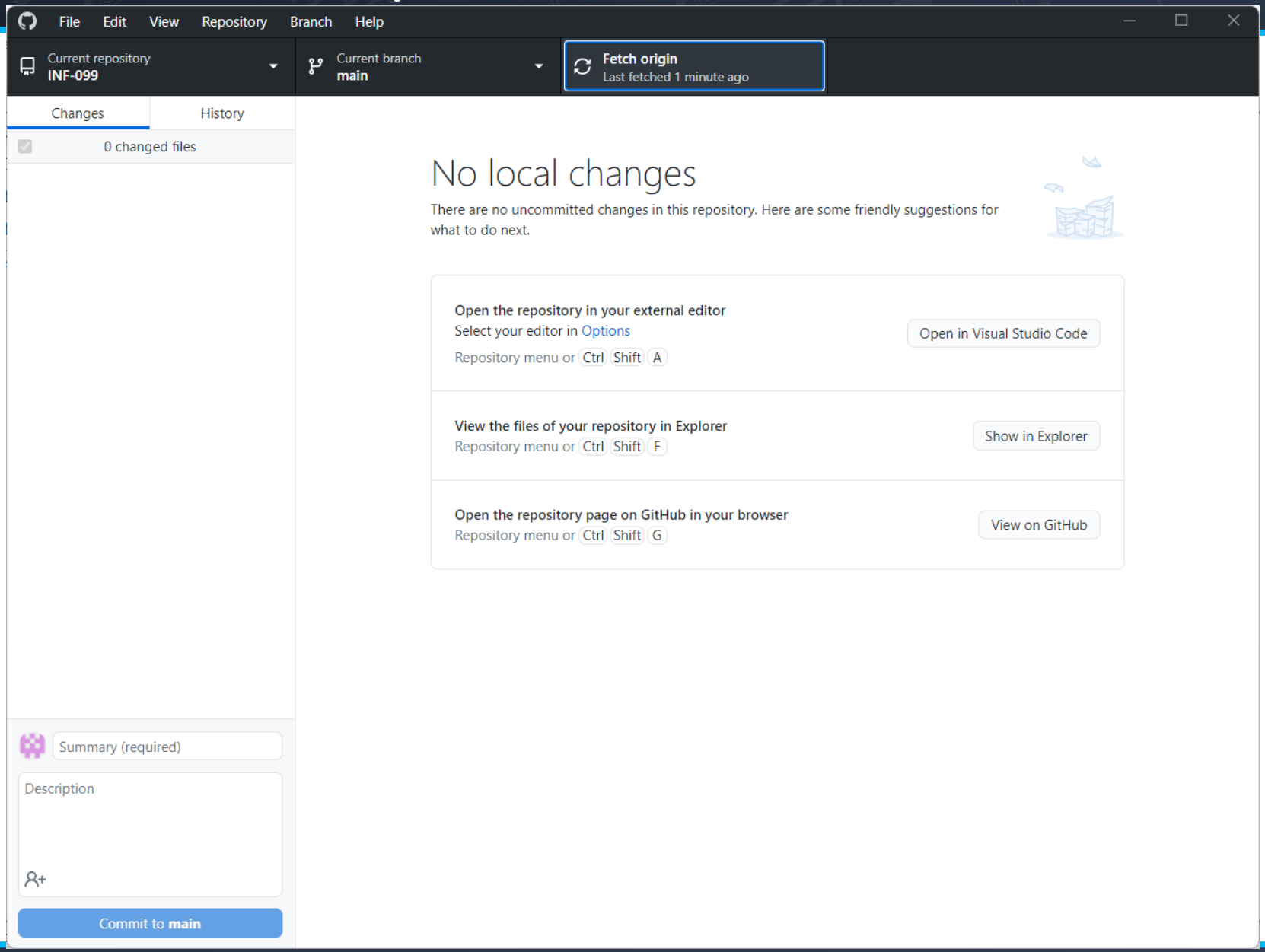
GitHub Desktop



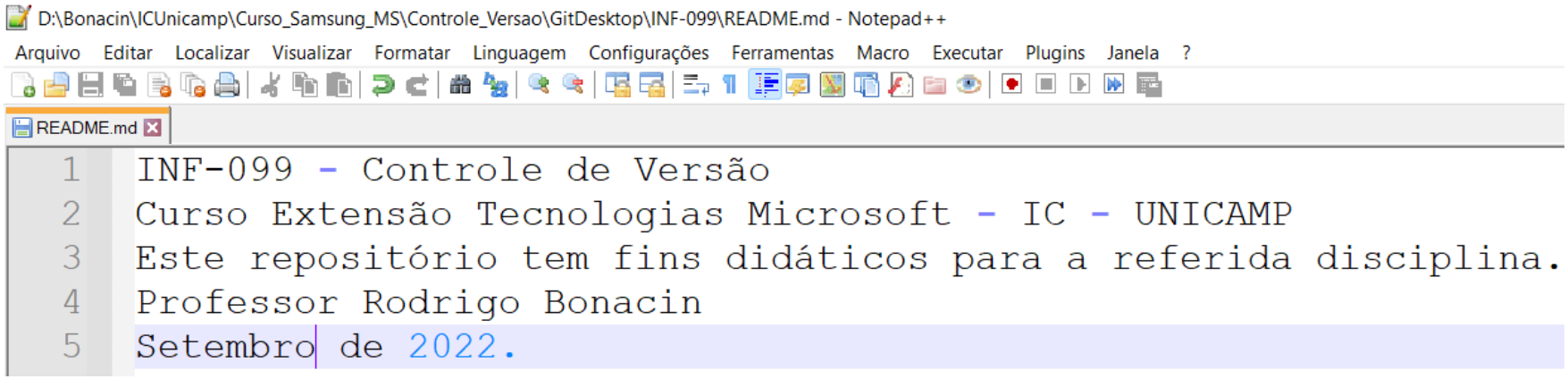
GitHub Desktop



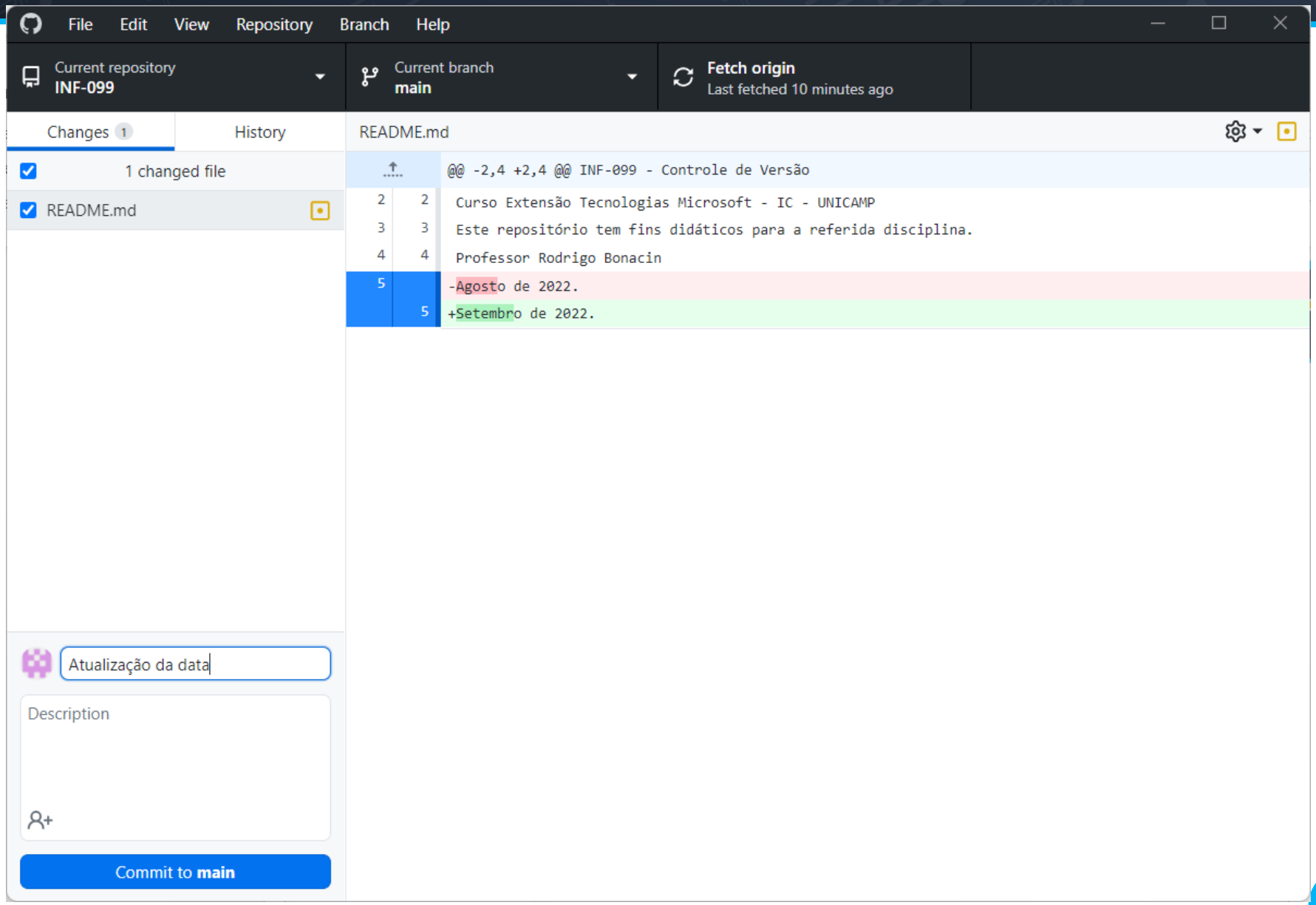
GitHub Desktop



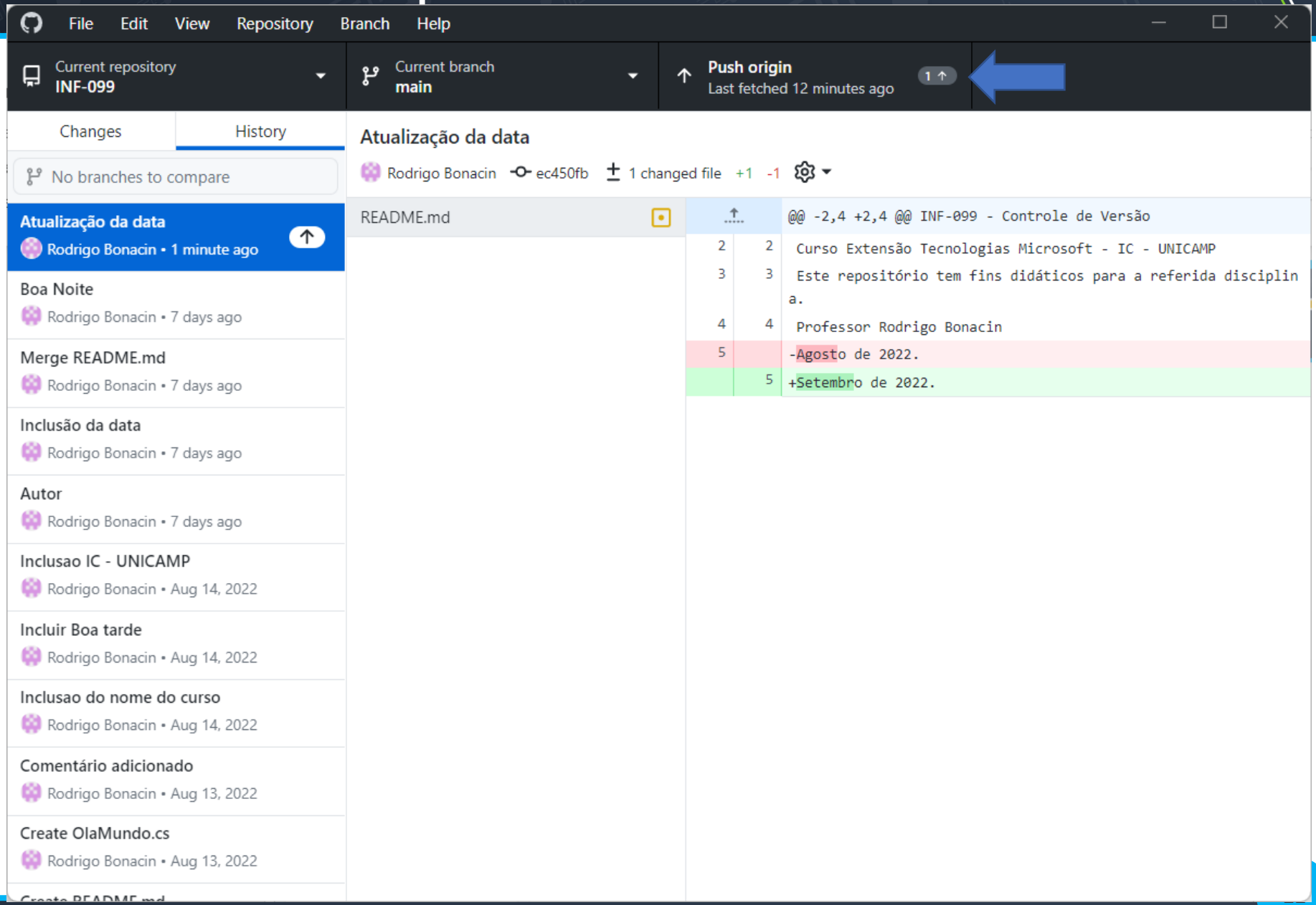
GitHub Desktop



GitHub Desktop



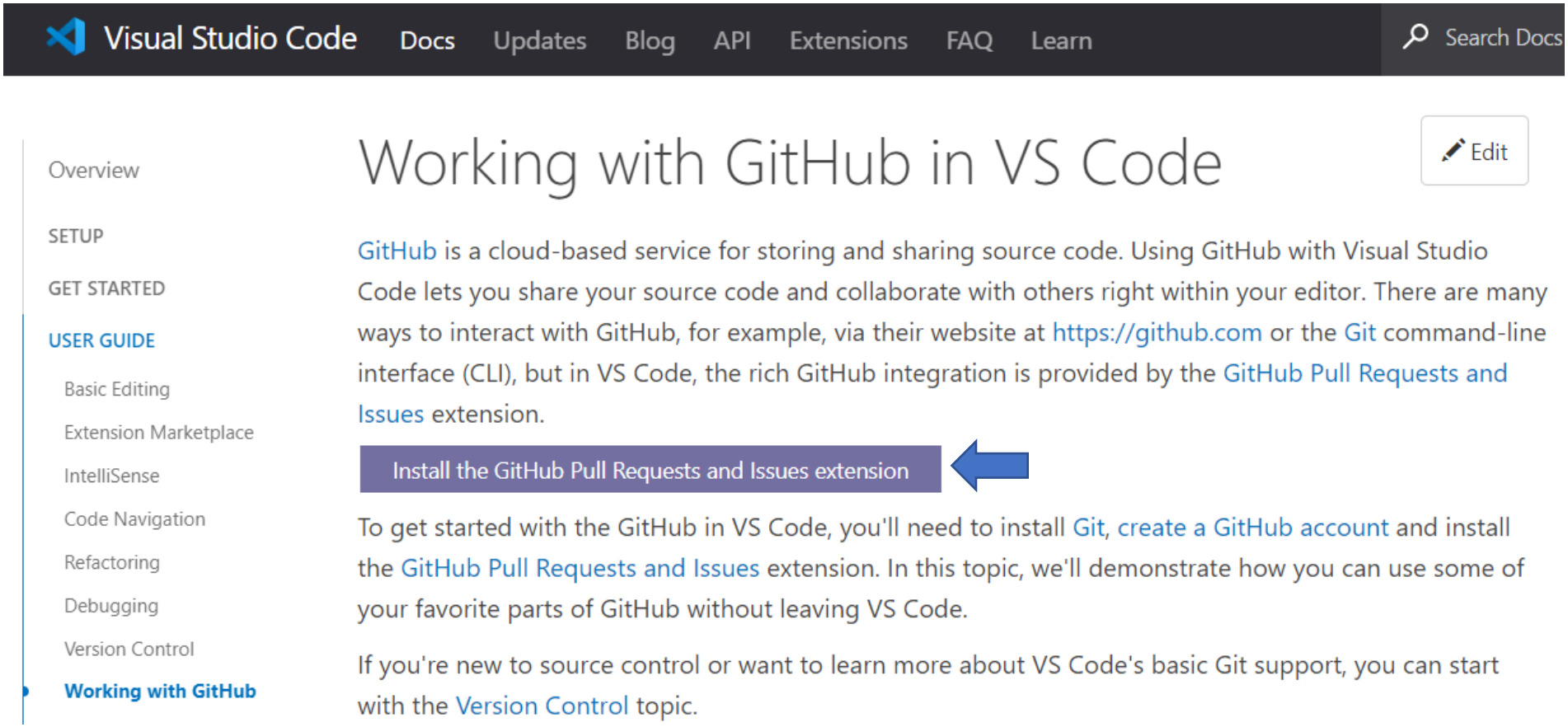
GitHub Desktop



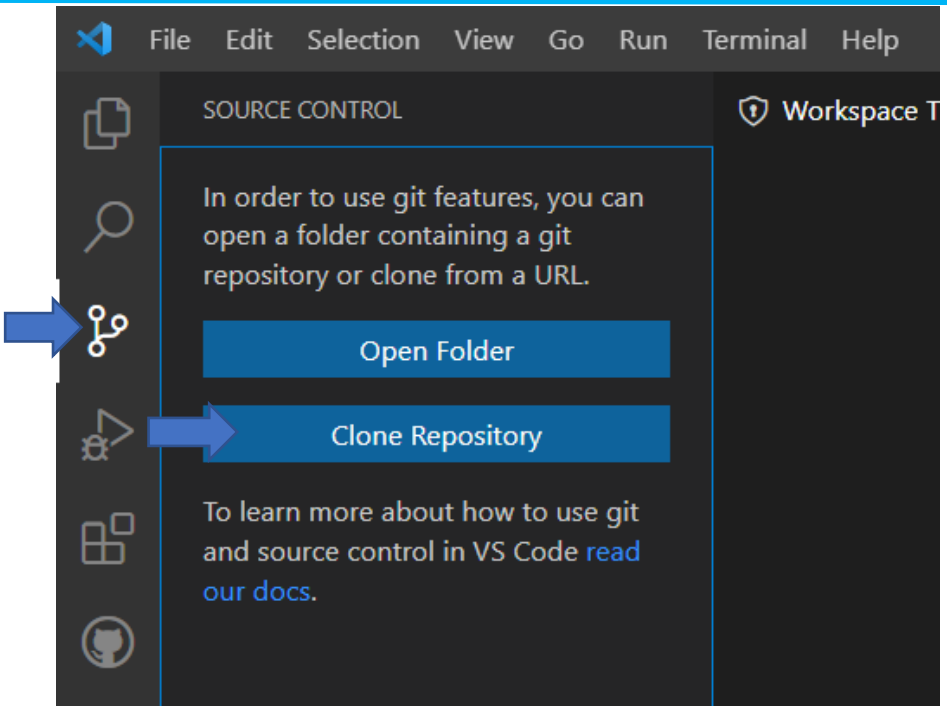
Ferramentas – VS Code + GitHub

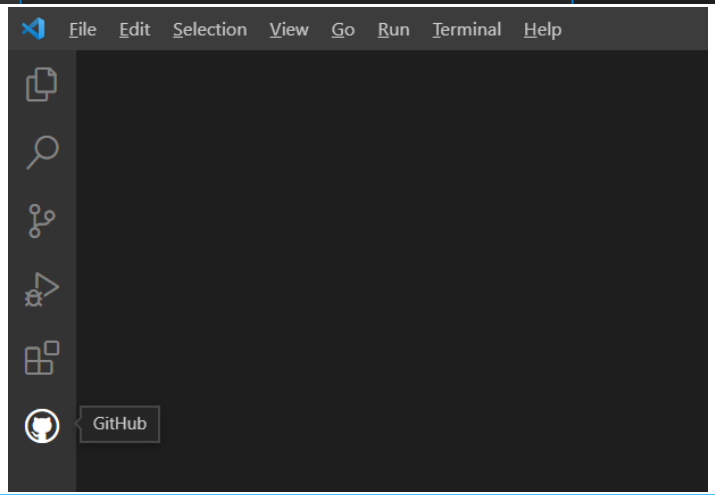
VS Code + GitHub

https://code.visualstudio.com/docs/editor/github

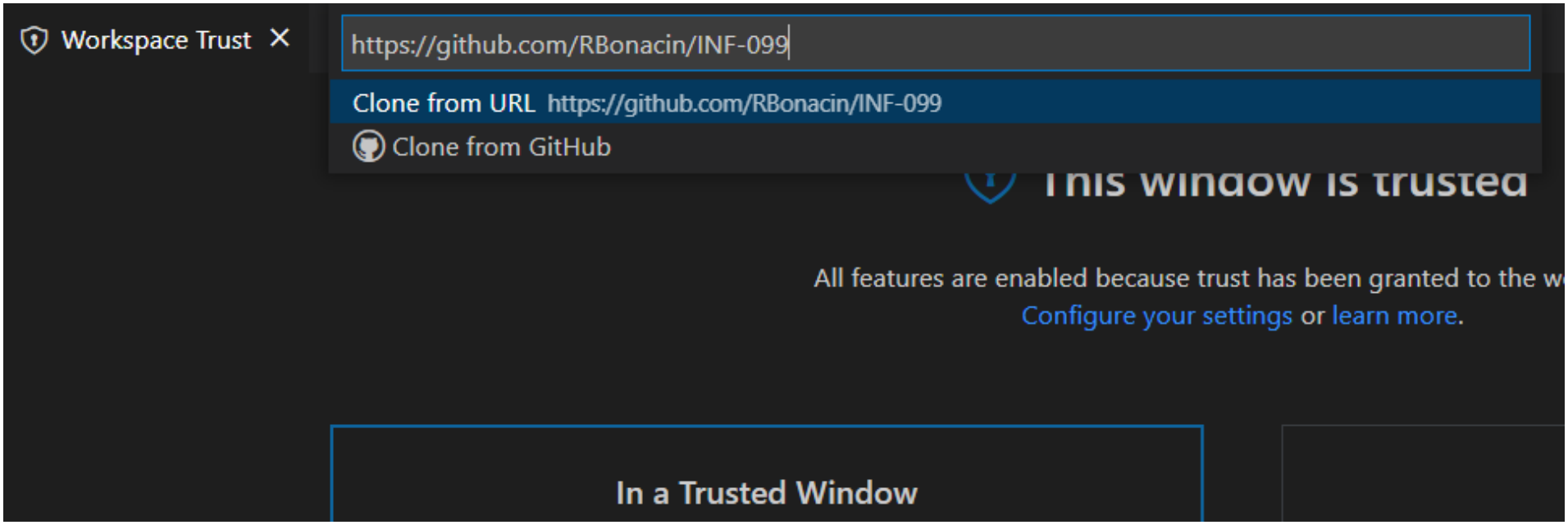


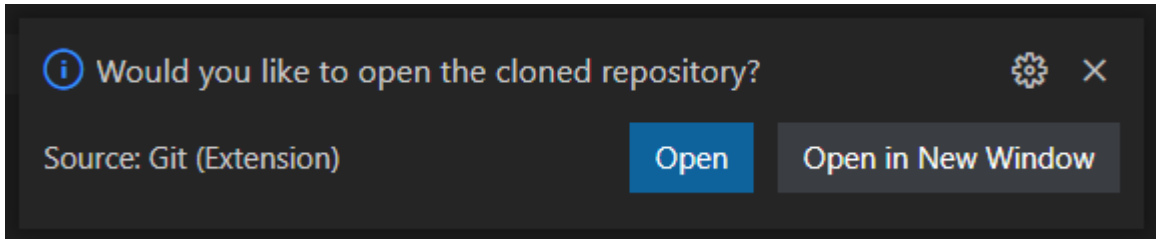
VS Code + GitHub



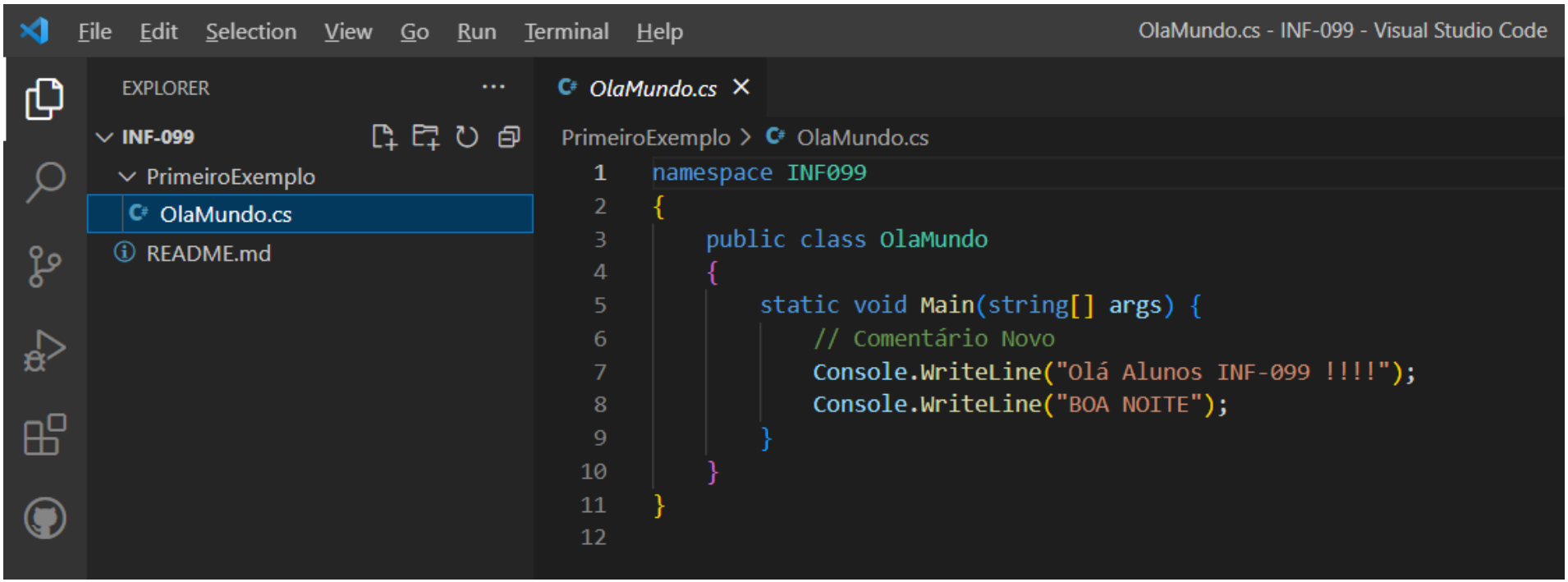


VS Code + GitHub

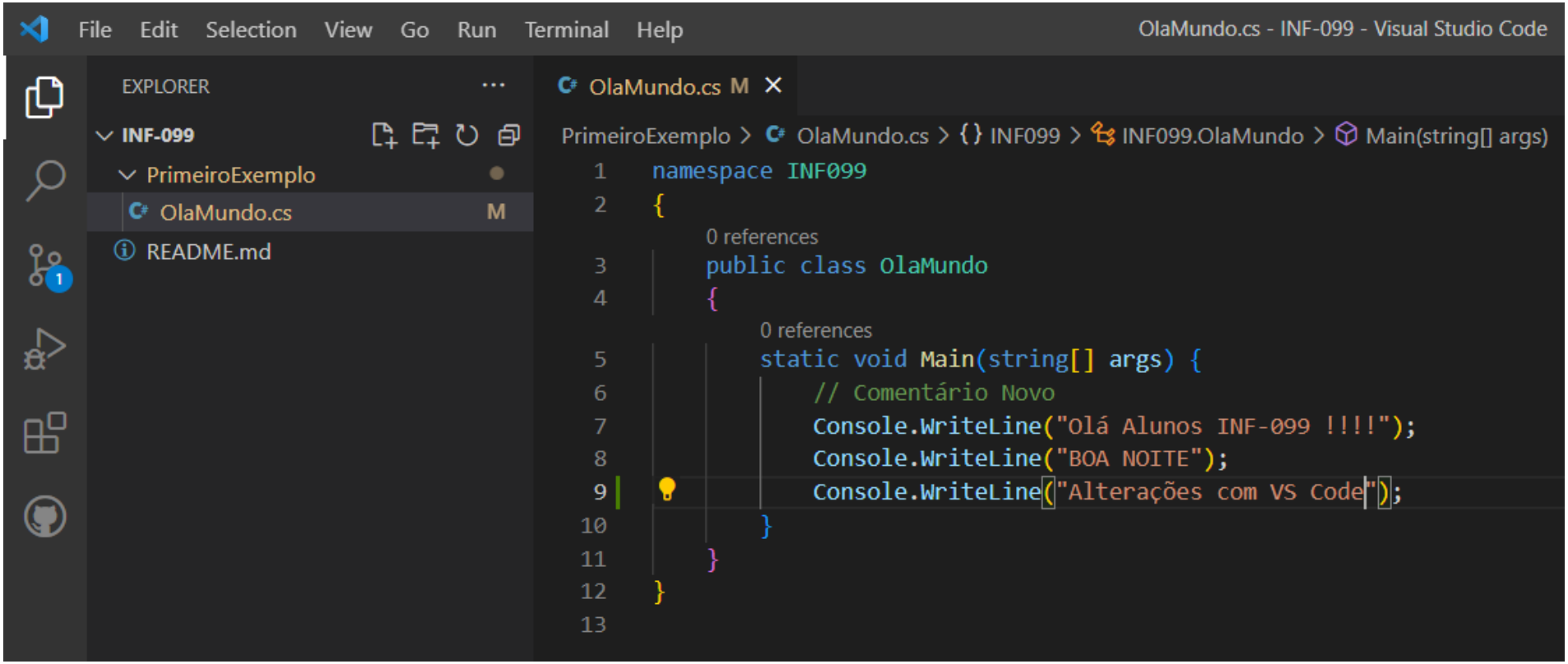




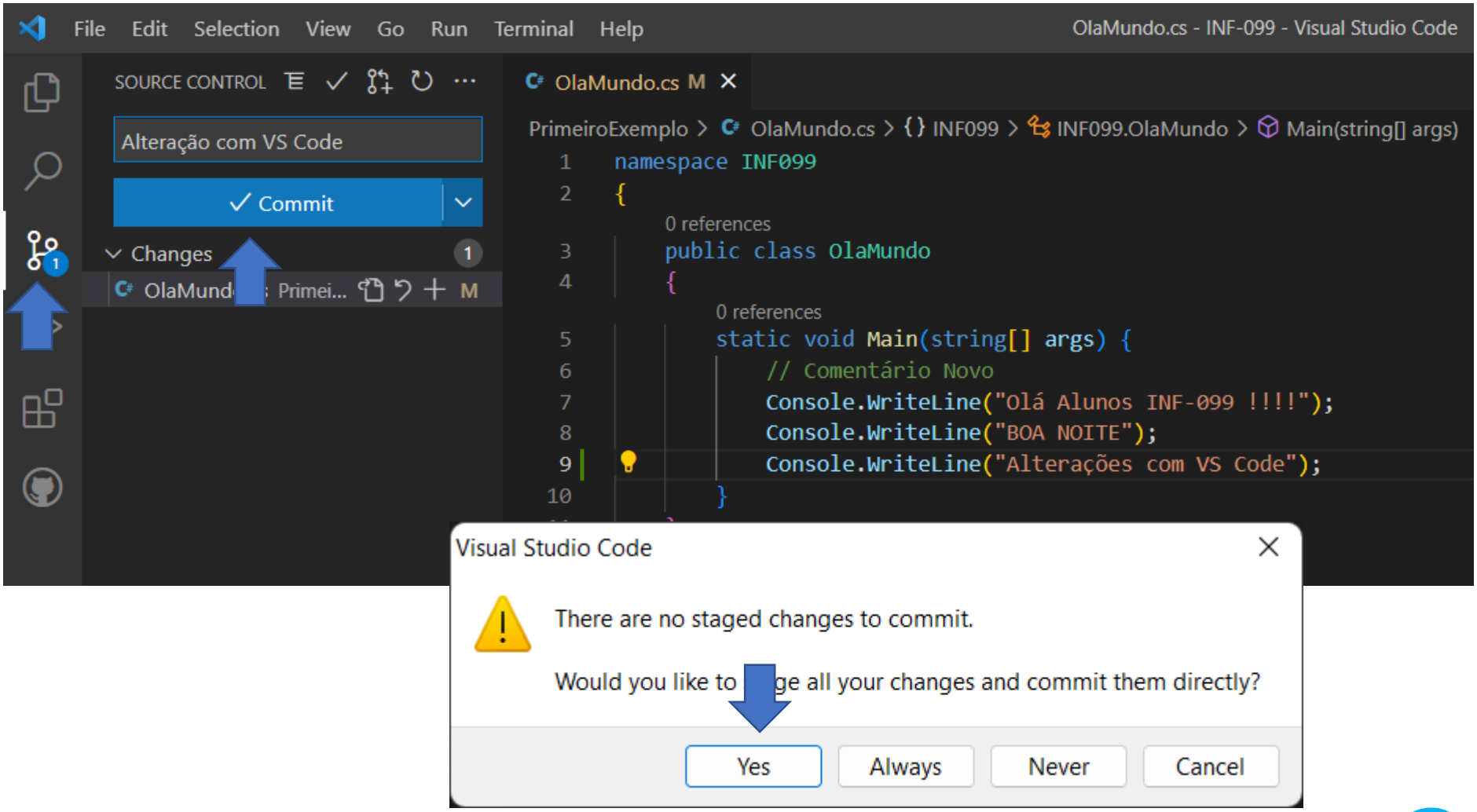
VS Code + GitHub



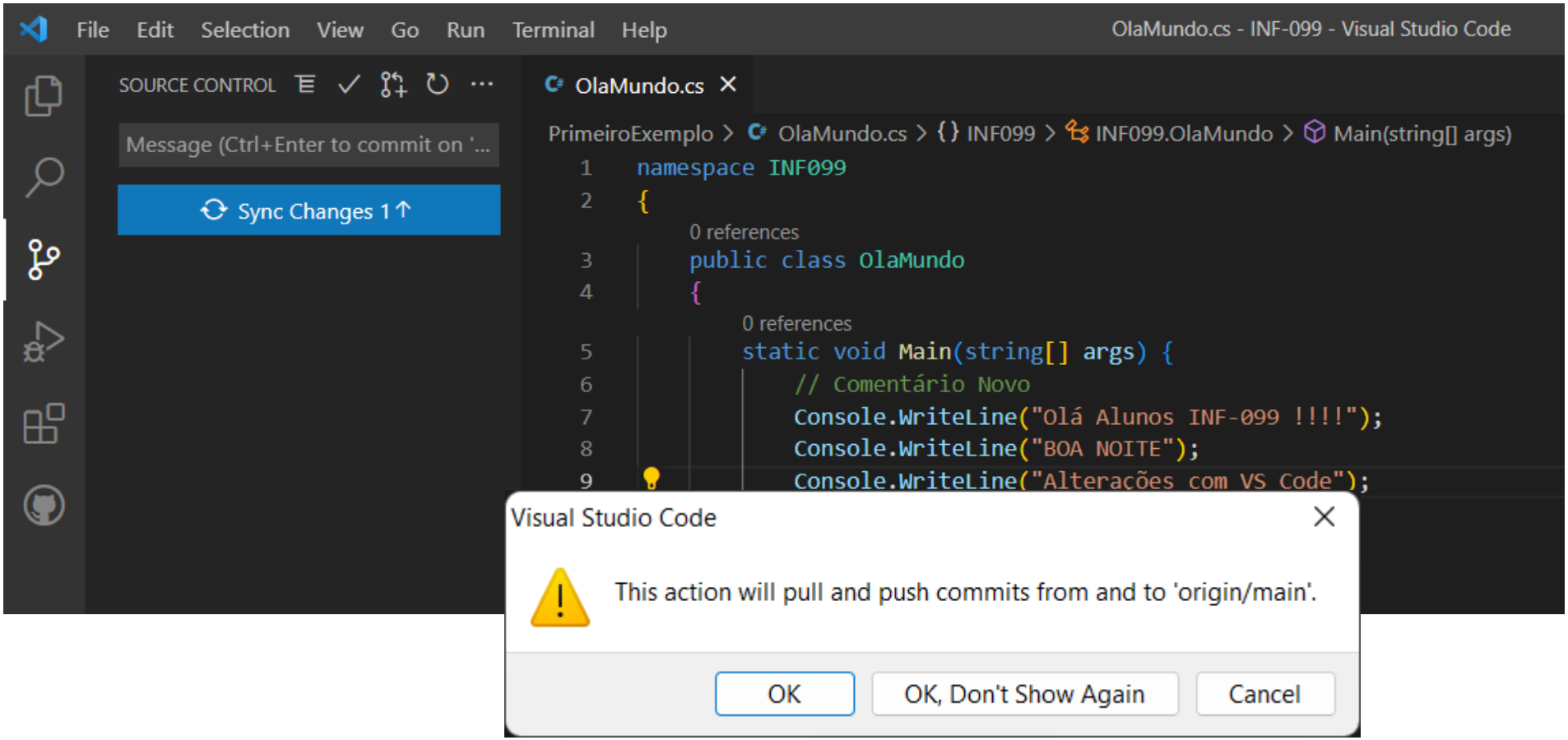
VS Code + GitHub



VS Code + GitHub



VS Code + GitHub



VS Code + GitHub

main ▾ INF-099 / PrimeiroExemplo / **OlaMundo.cs** / <> Jump to ▾

RBonacin Alteração com VS Code

1 contributor

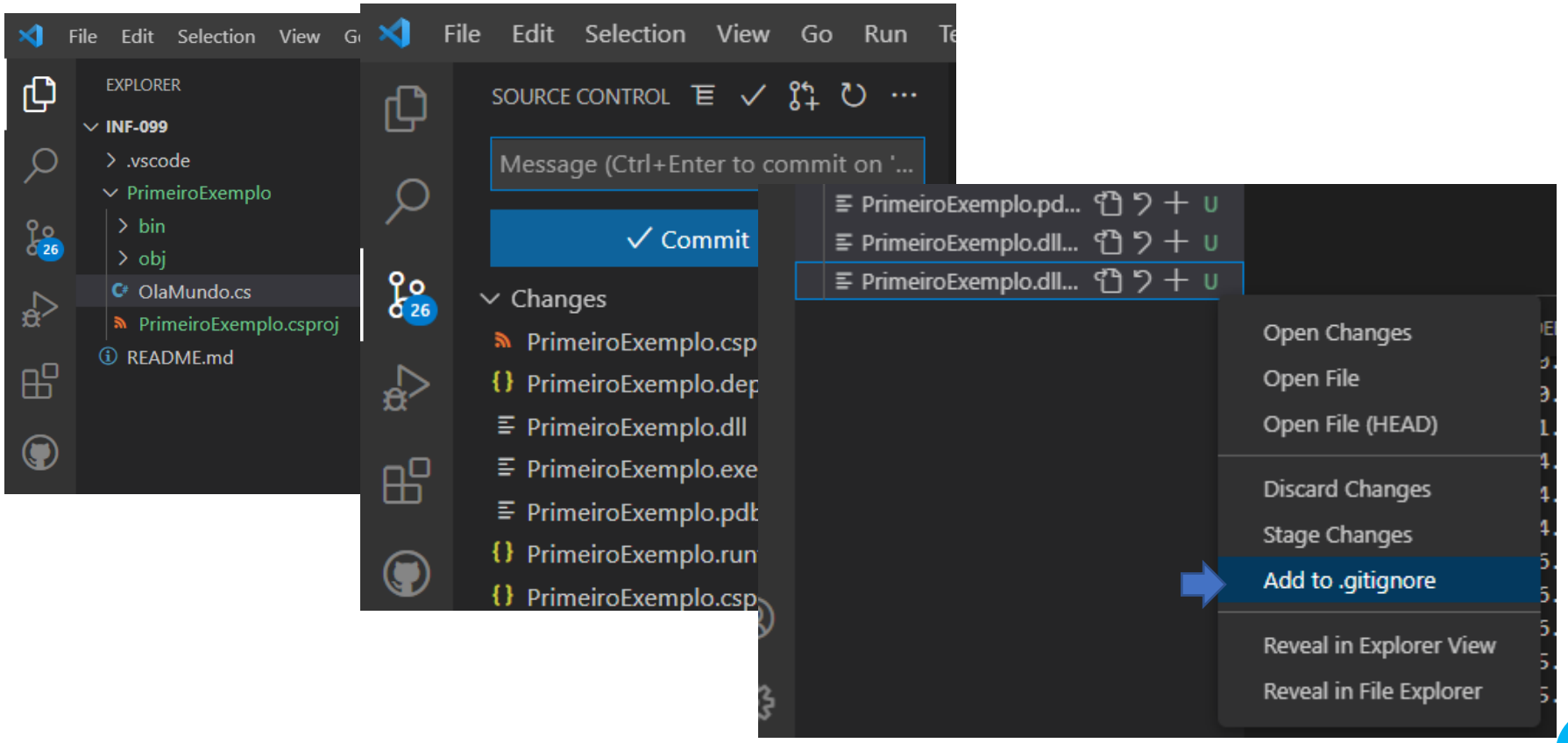
12 lines (12 sloc) | 287 Bytes

```
1   namespace INF099
2   {
3       public class OlaMundo
4       {
5           static void Main(string[] args) {
6                           // Comentário Novo
7               Console.WriteLine("Olá Alunos INF-099 !!!!");
8                           Console.WriteLine("BOA NOITE");
9               Console.WriteLine("Alterações com VS Code");
10          }
11      }
12  }
```

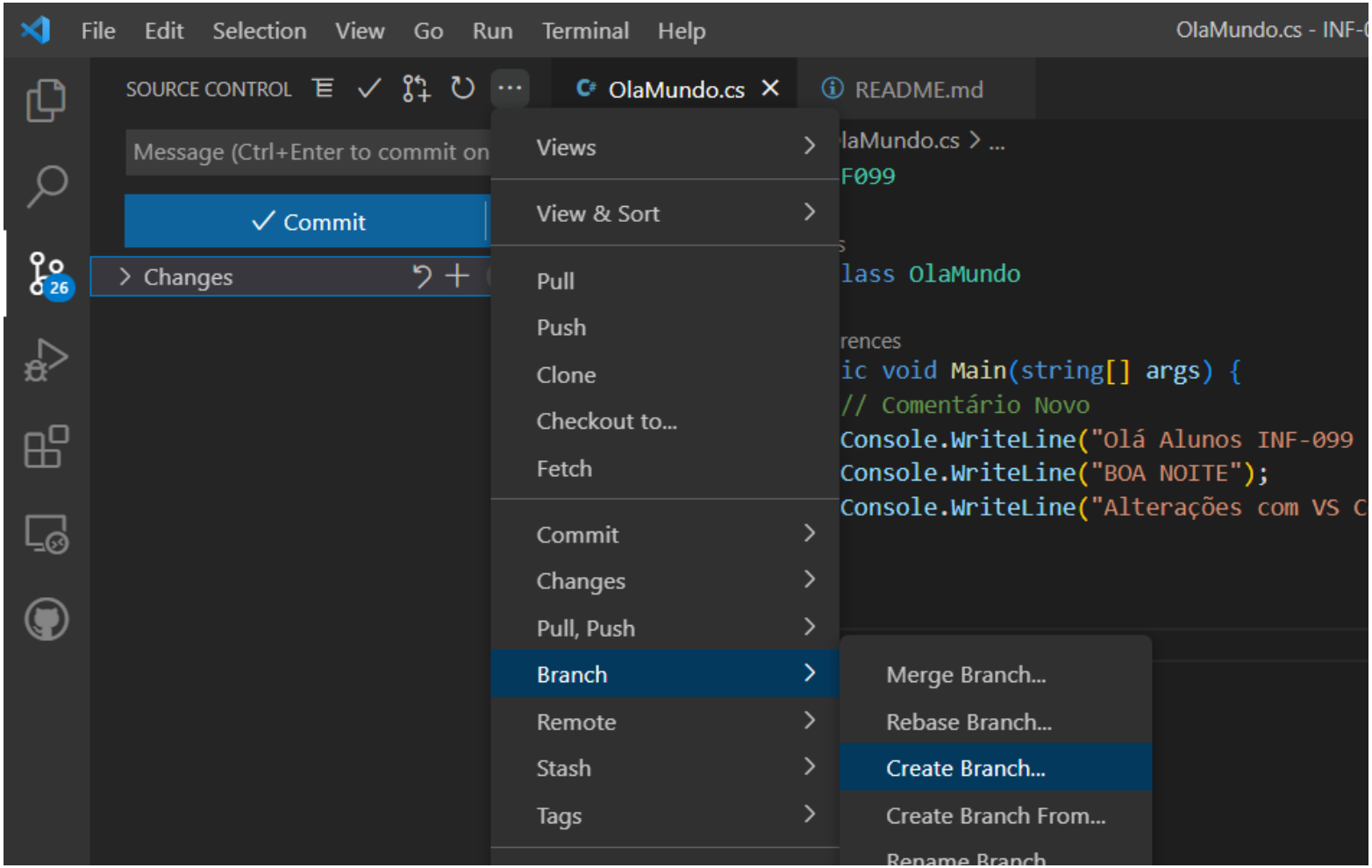
VS Code + GitHub

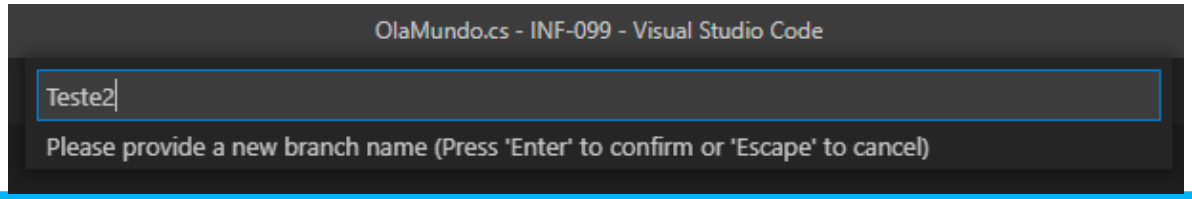
```
\INF-099\PrimeiroExemplo> dotnet new console --framework net6.0
```

```
PS D:\Bonacin\ICUnicamp\Curso_Samsung_MS\Controle_Versao\VSCode\INF-099\PrimeiroExemplo> dotnet run
Olá Alunos INF-099 !!!!
BOA NOITE
Alterações com VS Code
```

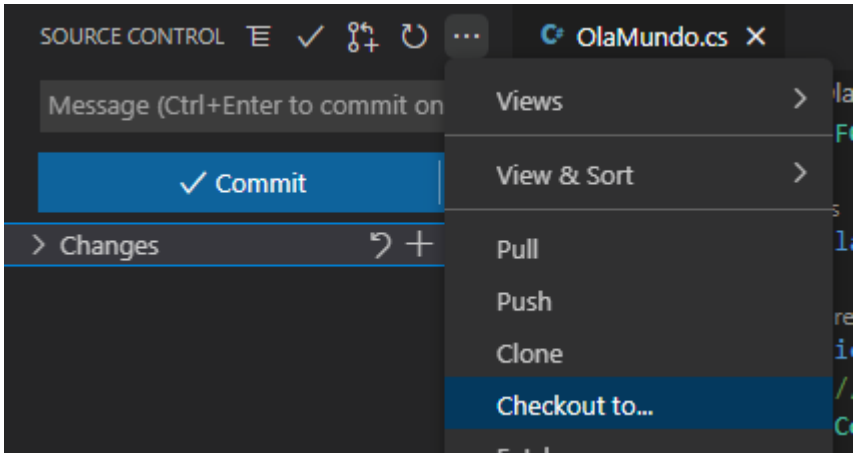


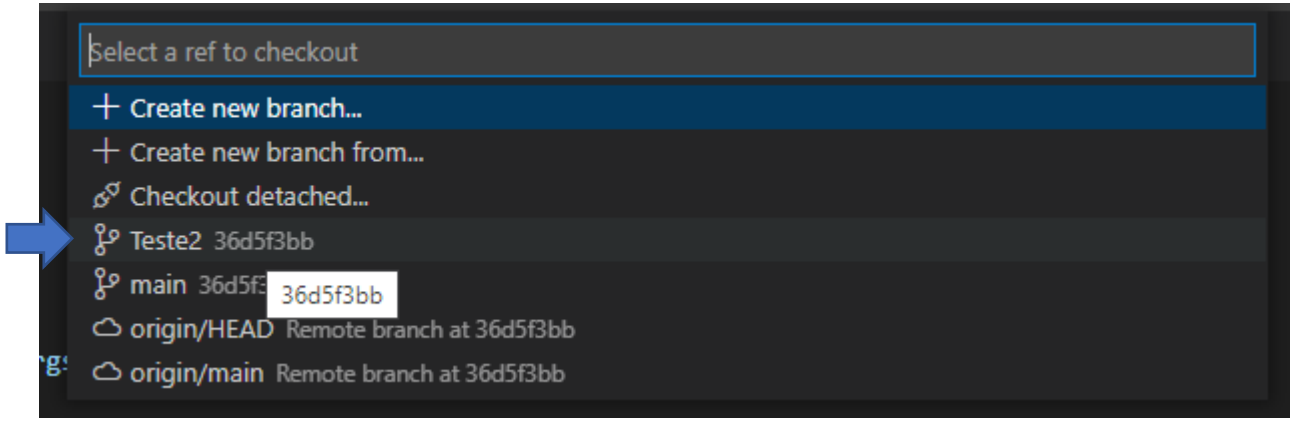
VS Code + GitHub



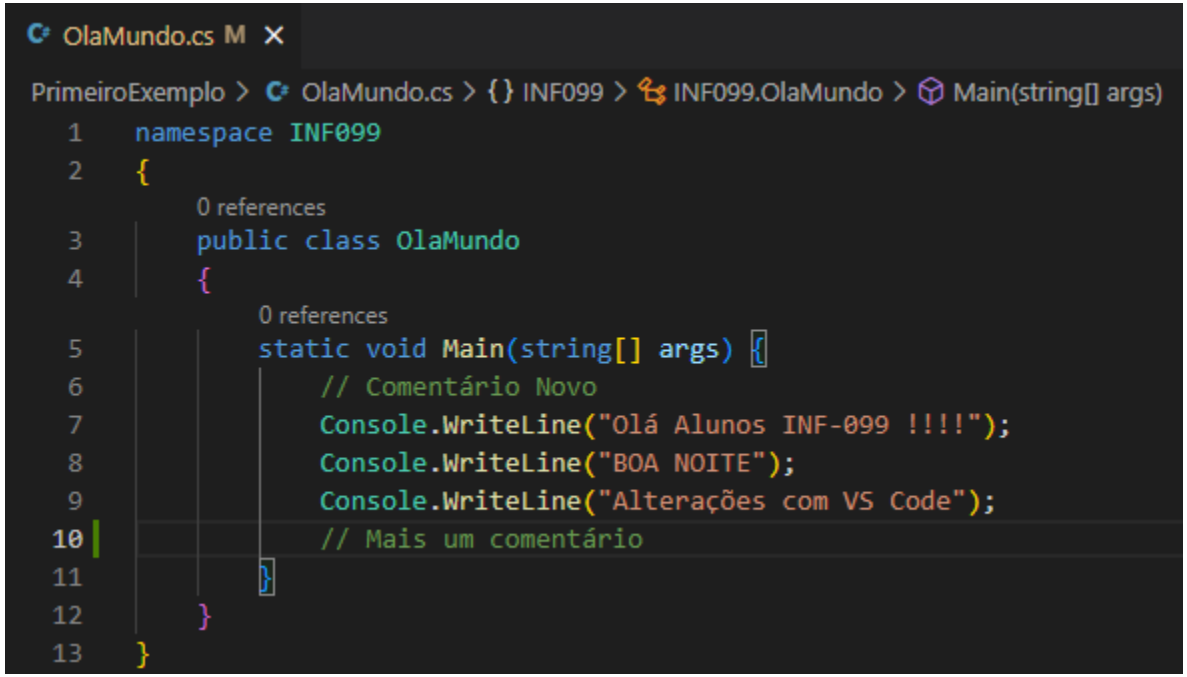


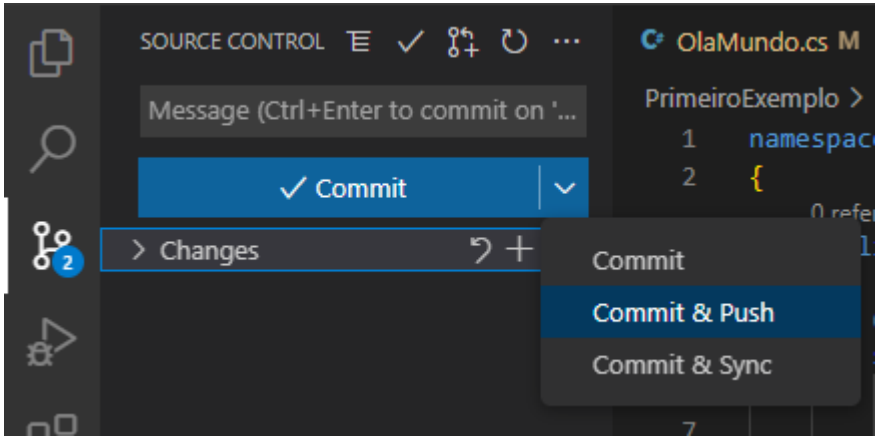
VS Code + GitHub

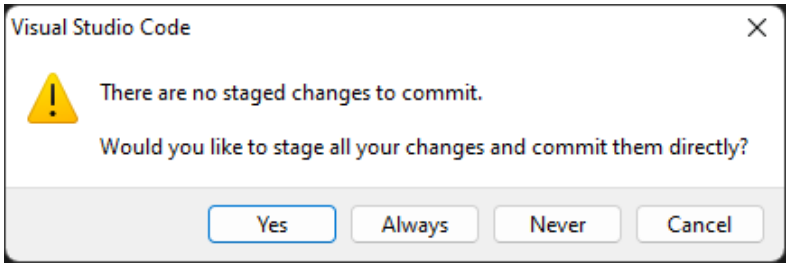




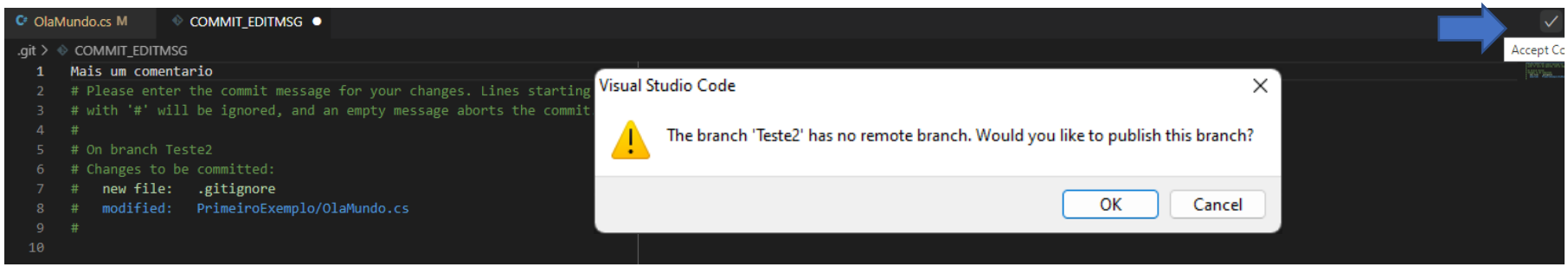
VS Code + GitHub

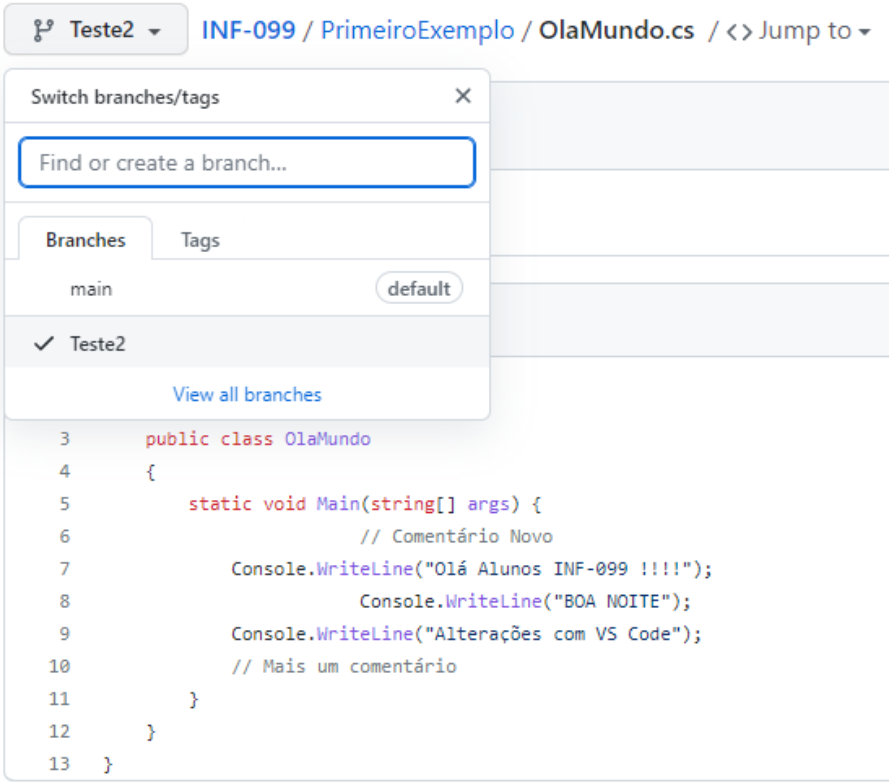




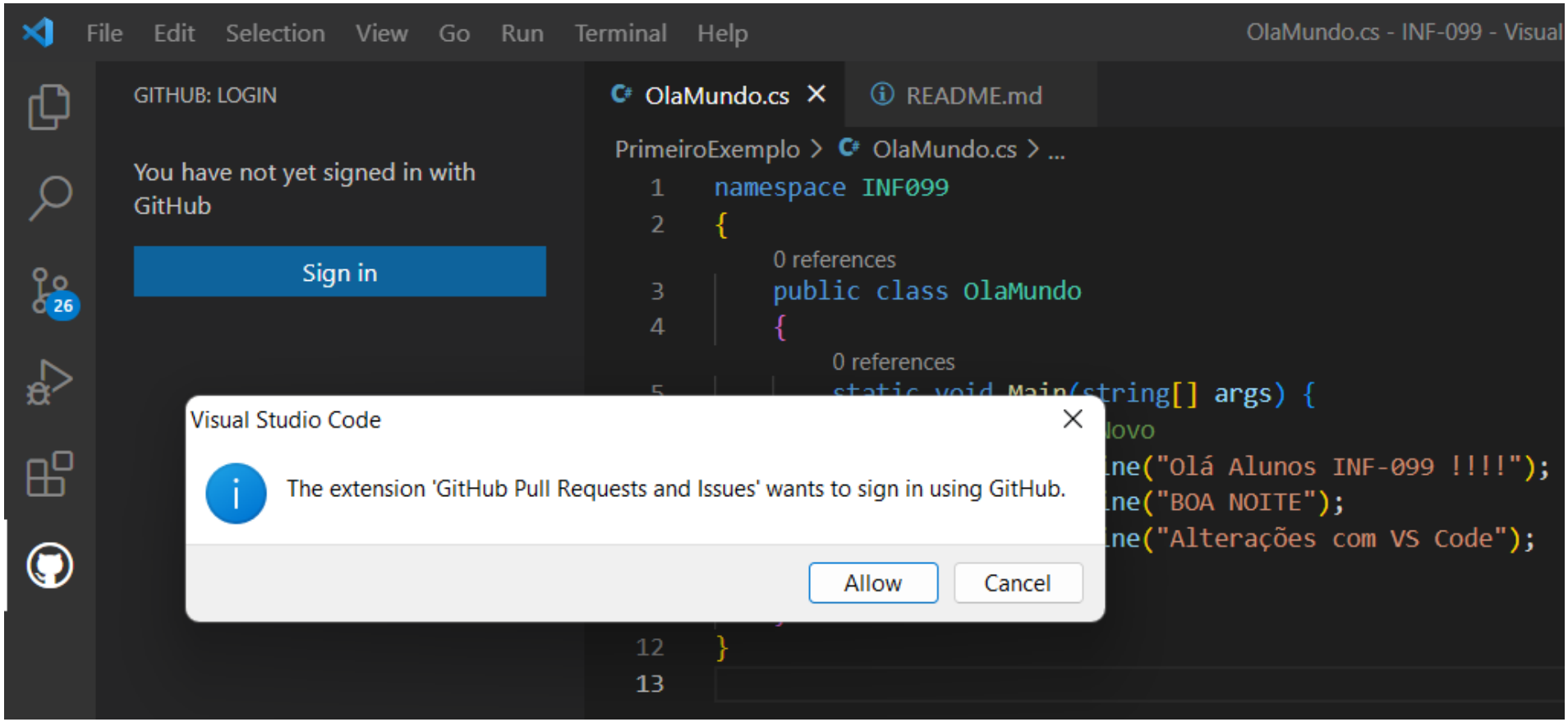


VS Code + GitHub

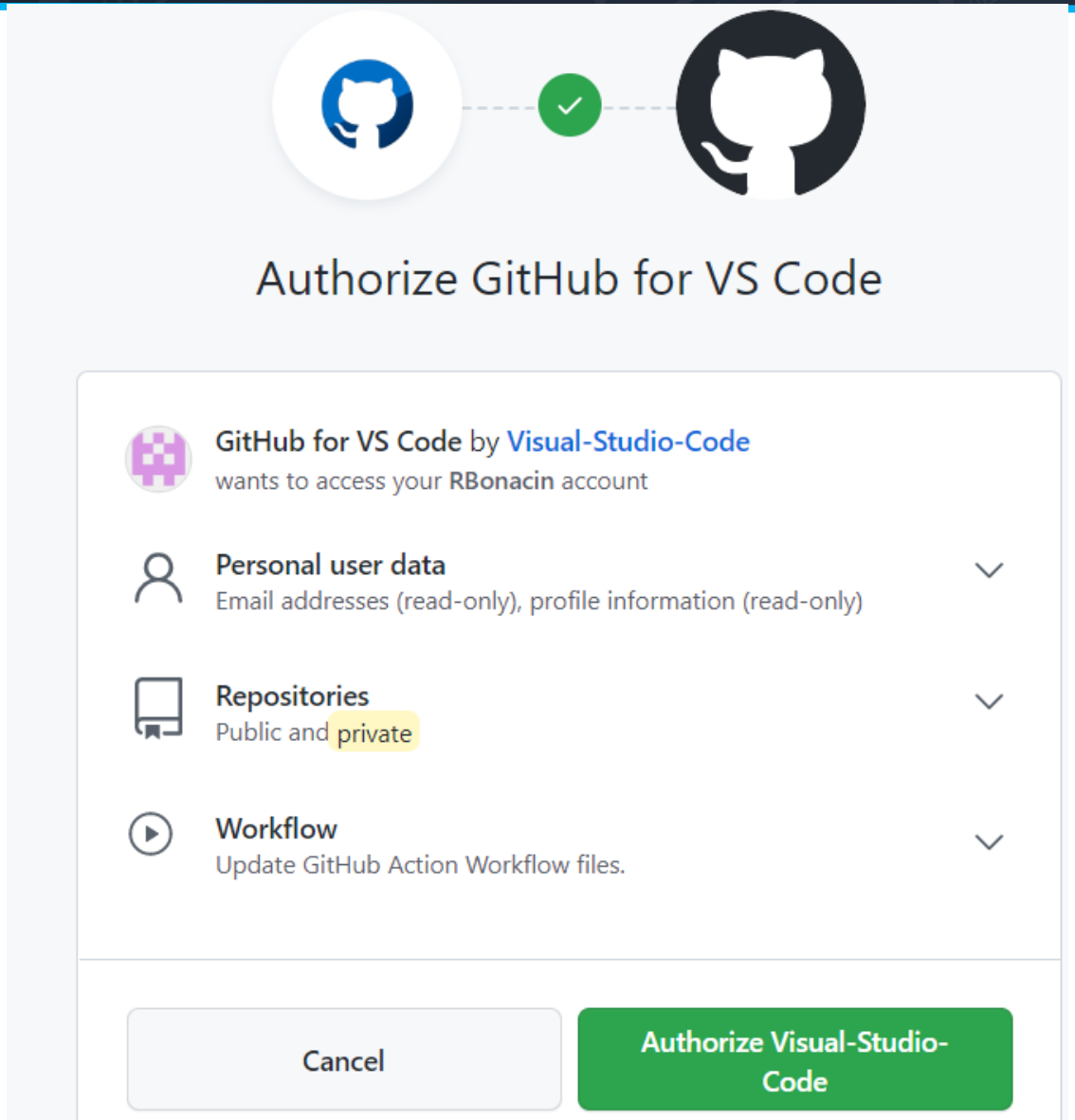




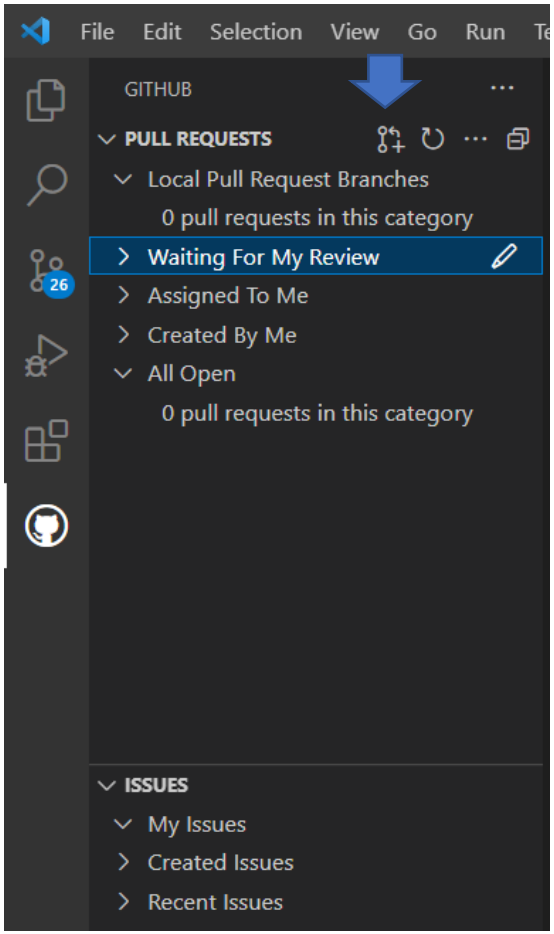
VS Code + GitHub

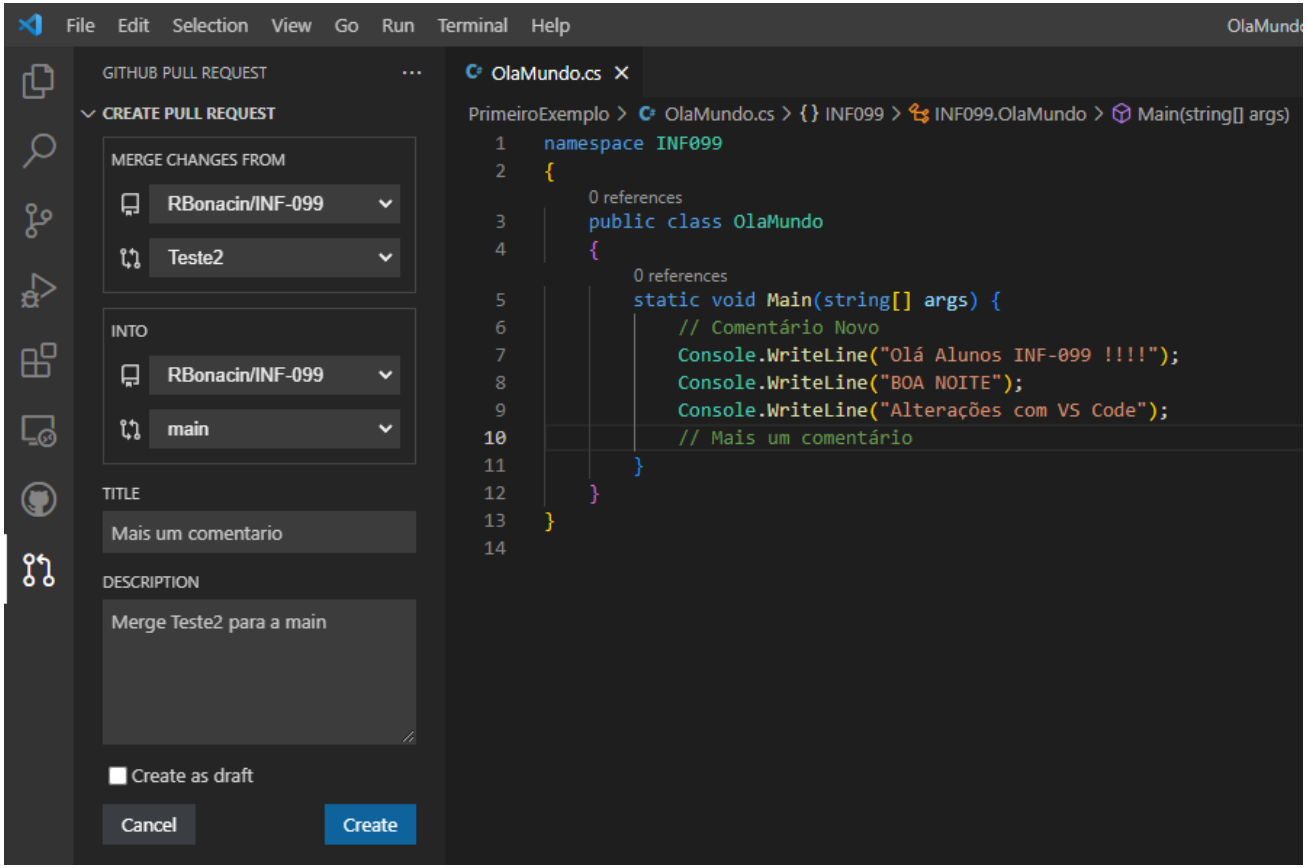


VS Code + GitHub

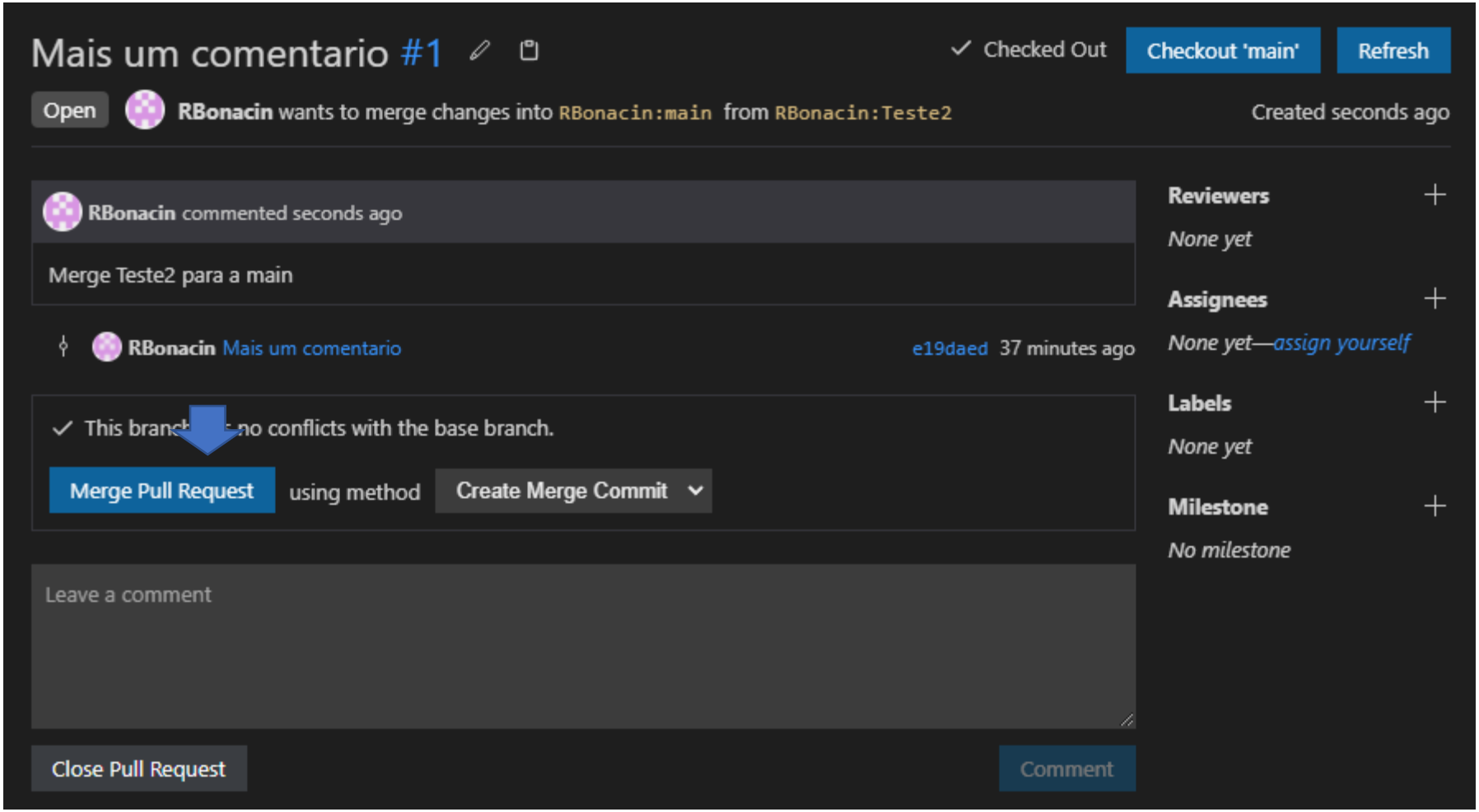


VS Code + GitHub





VS Code + GitHub



VS Code + GitHub

main ▾ INF-099 / PrimeiroExemplo / **OlaMundo.cs** / <> Jump to ▾

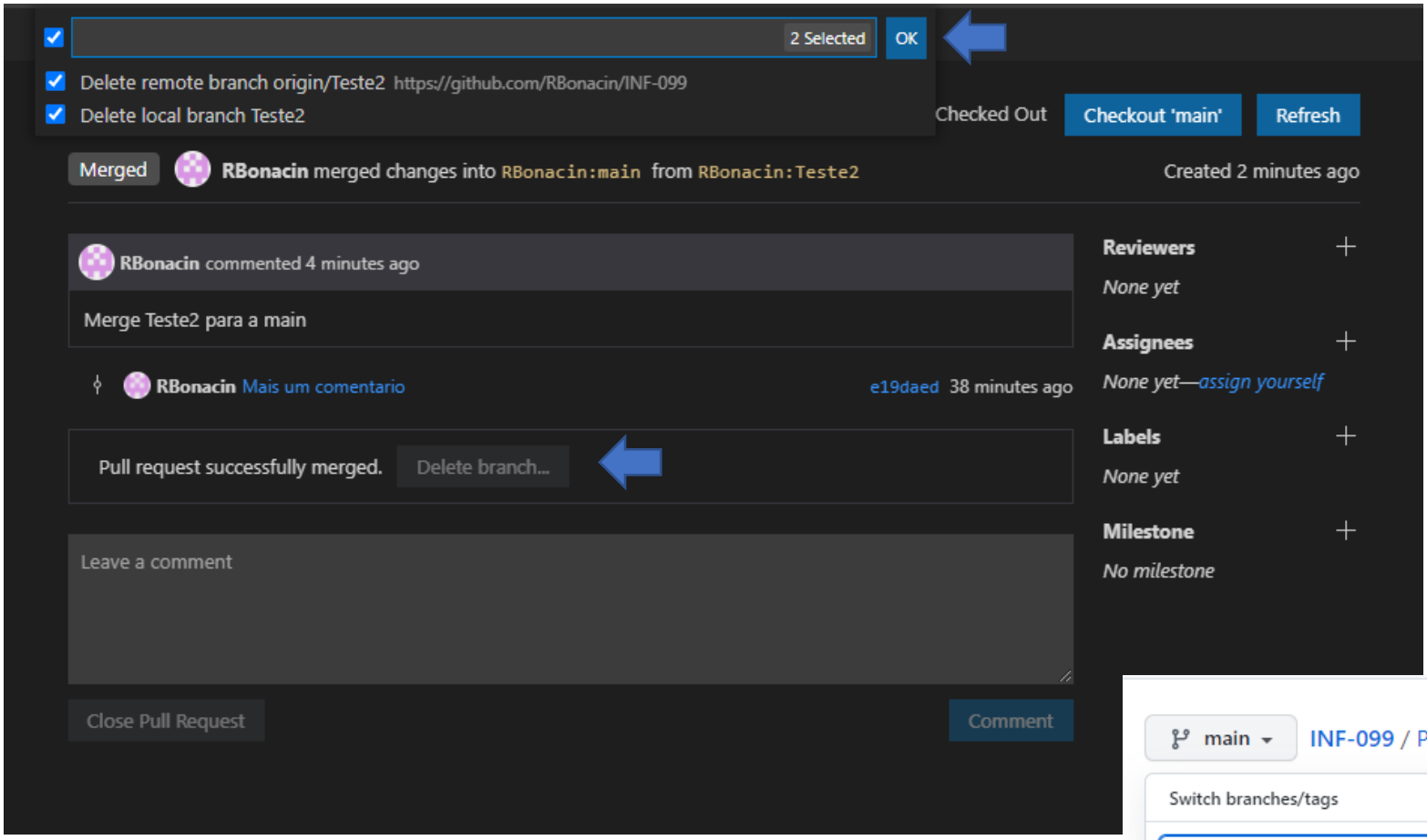
RBonacin Mais um comentario

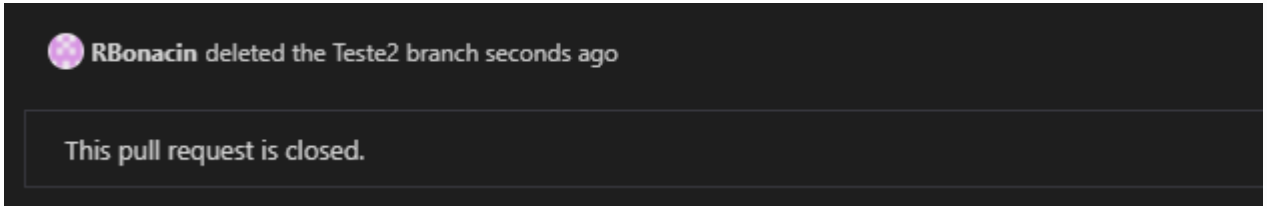
1 contributor

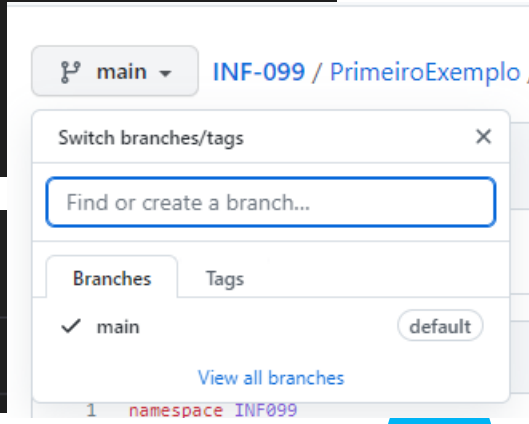
13 lines (13 sloc) | 322 Bytes

```
1   namespace INF099
2   {
3       public class OlaMundo
4       {
5           static void Main(string[] args) {
6                               // Comentário Novo
7               Console.WriteLine("Olá Alunos INF-099 !!!!");
8                               Console.WriteLine("BOA NOITE");
9               Console.WriteLine("Alterações com VS Code");
10              // Mais um comentário
11          }
12      }
13  }
```

VS Code + GitHub







Boas Práticas em Controle de Versão

Boas Práticas em CV

- Manter sempre seu repositório atualizado
  - Atualizar todos os arquivos antes de alterar ou utilizar algo
- Inserir comentários nos *commits*
- Não adicionar itens que não pertençam ao projeto
- Usar *branch* locais para fazer testes locais

Boas Práticas em CV

- Cada equipe com seu *branch*
  - Crie *branch* internas se necessário
- Não fazer *rebase* com o *branch* de outras equipe (ou membros)
- O *branch* "master/main" deve estar sempre funcional
  - *Features* novas (e não testadas) em *branchs* separados
  - E só depois integradas

Boas Práticas em CV

https://opensource.com/article/20/7/git-best-practices

- Formalize as convenções do Git para sua equipe
  - Instruções sobre boas práticas, convenções de nomes, instruções básicas de uso ...
- Mesclar as alterações corretamente
- *Rebase* sua ramificação com frequência
- *Squach* (vários em 1) os *commits* antes do merge
- Utilizar *Tags*
- Faça o software executável imprimir a *Tag*

Boas Práticas em CV

https://deepsource.io/blog/git-best-practices/

- Faça *commits* limpos e de propósito único
  - Tornar mais fácil para outras pessoas da equipe analisarem sua alteração, tornando as revisões de código mais eficientes.
  - Se o *commit* tiver que ser revertido completamente, é muito mais fácil fazê-lo
  - É simples acompanhar essas alterações com seu sistema de *ticketing*.

Boas Práticas em CV

https://deepsource.io/blog/git-best-practices/

- Escreva mensagens de *commit* significativas

```
feat: add beta sequence

^--^  ^---------------^

|     |

|     +-> Summary in present tense.

|

+-------> Type: chore, docs, feat, fix, refactor, style, or test.
```

  - Se estiver usando um sistema de *ticket* (controle de incidência, configuração, mudança ...), incluir também o *id* do ticket na descrição

Boas Práticas em CV

https://deepsource.io/blog/git-best-practices/

- Faça *commit* sempre, faça *commit* com frequência
  - Em vez de esperar para fazer o *commit* perfeito, é melhor trabalhar em pequenos pedaços e continuar confirmando seu trabalho

    (não é fazer *commit* de versões instáveis)
  - Se você estiver trabalhando em um *branch* que pode levar algum tempo para ser concluído, isso ajuda a manter seu código atualizado com as alterações mais recentes para evitar conflitos

Boas Práticas em CV

https://deepsource.io/blog/git-best-practices/

- Não altere o histórico publicado
  - Uma vez que um *commit* tenha sido mesclado em um *branch* padrão (e esteja visível para outros), é altamente recomendável não alterar o histórico
  - Embora o *rebase* seja um recurso útil, ele deve ser usado apenas em *branches* com os quais você está trabalhando

Boas Práticas em CV

https://deepsource.io/blog/git-best-practices/

- Não faça *commit* de arquivos gerados
  - Normalmente, apenas fazemos *commit* (controlamos) arquivos que exigiram esforço manual, ou seja, que não podem ser regerados automaticamente
  - É útil adicionar um arquivo *.gitignore* na raiz do seu repositório para informar automaticamente ao Git quais arquivos ou caminhos você não deseja rastrear (ex: *.dll, *.exe ...)

Boas Práticas em CV

https://sourcelevel.io/blog/7-git-best-practices-to-start-using-in-your-next-commit

1. Não faça *git push* direto no *master*. Ramifique-o!
   - Enviar o código diretamente para o *master* não promove a colaboração
   - O Git simplifica a comparação de código entre dois *branchs*
     - Isso pode gerar discussões saudáveis, melhorar a qualidade do código base e propagar conhecimento entre os desenvolvedores

Boas Práticas em CV

https://sourcelevel.io/blog/7-git-best-practices-to-start-using-in-your-next-commit

2. Configure adequadamente a autoria do *commit*

- Você deve pelo menos definir seu nome e endereço de e-mail corretamente

- *Commits* são ferramentas de comunicação, saber quem fez uma alteração específica pode ajudá-lo no futuro

```
D:\Bonacin\ICUnicamp\Curso_Samsung_MS\Controle_Versao\MeuProjeto\INF-099>git blame README.md
^9e3098f (Rodrigo Bonacin 2022-08-12 16:43:13 -0300 1) INF-099 - Controle de Versão
22d99455 (Rodrigo Bonacin 2022-08-14 12:22:39 -0300 2) Curso Extensão Tecnologias Microsoft - IC - UNICAMP
^9e3098f (Rodrigo Bonacin 2022-08-12 16:43:13 -0300 3) Este repositório tem fins didáticos para a referida disciplina.
782cdd6a (Rodrigo Bonacin 2022-08-14 18:13:18 -0300 4) Professor Rodrigo Bonacin
2f2d5695 (Rodrigo Bonacin 2022-08-14 19:19:47 -0300 5) Agosto de 2022.
```

- Mas não use “blame” para culpar

Boas Práticas em CV

https://sourcelevel.io/blog/7-git-best-practices-to-start-using-in-your-next-commit

3. Escreva mensagens de confirmação descritivas e significativas

https://cbea.ms/git-commit/

Boas Práticas em CV

https://sourcelevel.io/blog/7-git-best-practices-to-start-using-in-your-next-commit

4. Faça *commit* apenas de trabalhos relacionados

- Faça *commit* de uma quantidade mínima de linhas que façam sentido juntas
  - O *commit* não deve ser de *x* e *y* e *z* (que não são relacionados)
  - *single-responsibility principle*

Boas Práticas em CV

https://sourcelevel.io/blog/7-git-best-practices-to-start-using-in-your-next-commit

5. Evite reescrever a história do *branch mestre*

- *Git rebase* é útil, mas usar git rebase em *branches* publicados, como o master, pode gerar muitos conflitos para os outros contribuidores
- Em *git push –force* , sempre especificar o branch destino para evitar erros
  - Evite em qualquer *branch* publicado

Boas Práticas em CV

https://sourcelevel.io/blog/7-git-best-practices-to-start-using-in-your-next-commit

6. *Rebase* seu *branch* de trabalho com frequência

- É crucial sempre manter seu *branch rebase* com o código mais recente

- Você deve fazer *rebase* para evitar bugs, retrabalho e o trabalho tedioso de resolver conflitos com o *branch upstream* (remoto)

- `git checkout <upstream_branch>`
- `git pull`
- `git checkout -`
- `git rebase <upstream_branch>`

Boas Práticas em CV

https://sourcelevel.io/blog/7-git-best-practices-to-start-using-in-your-next-commit

7. Conheça a ferramenta. Não tenha medo de usá-la.

- O git tem diversos comandos úteis que vão além
  - Veja: *git help –a*
  - Ex:
    - `git cherry-pick`
    - `git diff` and `git apply`
    - `git stash`
    - `git bisect`

Boas Práticas em CV

https://www.git-tower.com/learn/git/ebook/en/command-line/appendix/best-practices

a) Fazer *commit* de alterações relacionadas

...

b) Fazer *commit* frequentemente

...

c) Não fazer *commit* de trabalho pela metade

...

d) Teste antes de fazer *commit*

...

e) Escreva boas mensagens de commit

...

Boas Práticas em CV

https://www.git-tower.com/learn/git/ebook/en/command-line/appendix/best-practices

f) O controle de versão não é um sistema de backup

- Fazer backup de seus arquivos em um servidor remoto é um bom efeito colateral de ter um sistema de controle de versão
- Mas você não deve usar seu VCS como se fosse um sistema de backup
- Ao fazer o controle de versão, você deve prestar atenção a semântica do *commit* (veja “mudanças relacionadas”) – você não deve apenas amontoar arquivos

Boas Práticas em CV

https://www.git-tower.com/learn/git/ebook/en/command-line/appendix/best-practices

g) Utilizar *Branches*

- É fácil de usar, é um requisito básico do git ...

h) Concordar em um fluxo de trabalho

- O Git permite que você escolha entre muitos fluxos de trabalho diferentes: *branches* de longa duração, *branches* de tópicos, *merge* ou *rebase*, *git-flow*...
- A escolha depende de alguns fatores:
  - Seu projeto, seus fluxos de trabalho gerais de desenvolvimento e implantação e (talvez o mais importante) das preferências pessoais suas e de seus colegas de equipe
- Seja como for que você escolha trabalhar, apenas certifique-se de concordar com um fluxo de trabalho comum que todos seguem

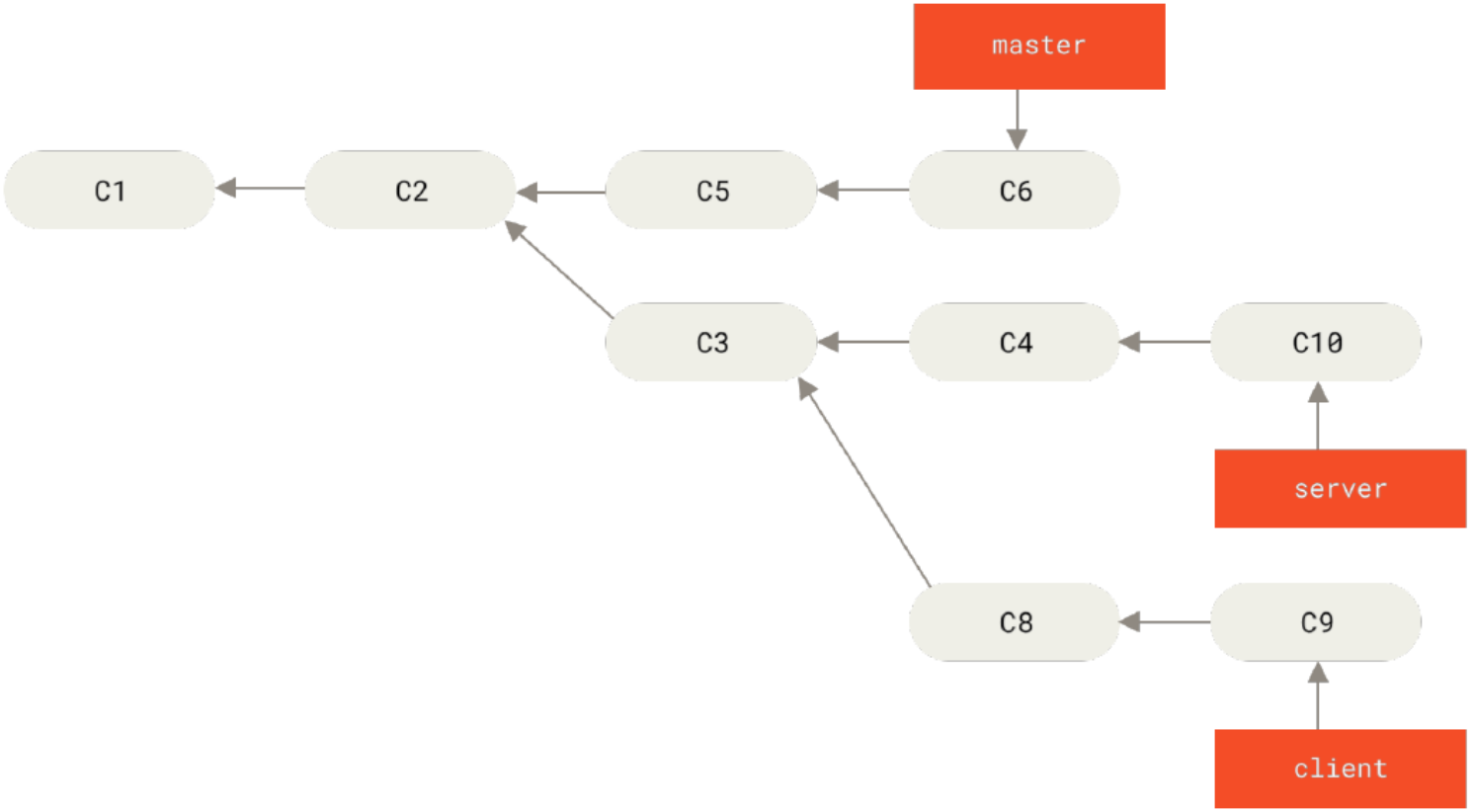
Boas Práticas em CV

. . .

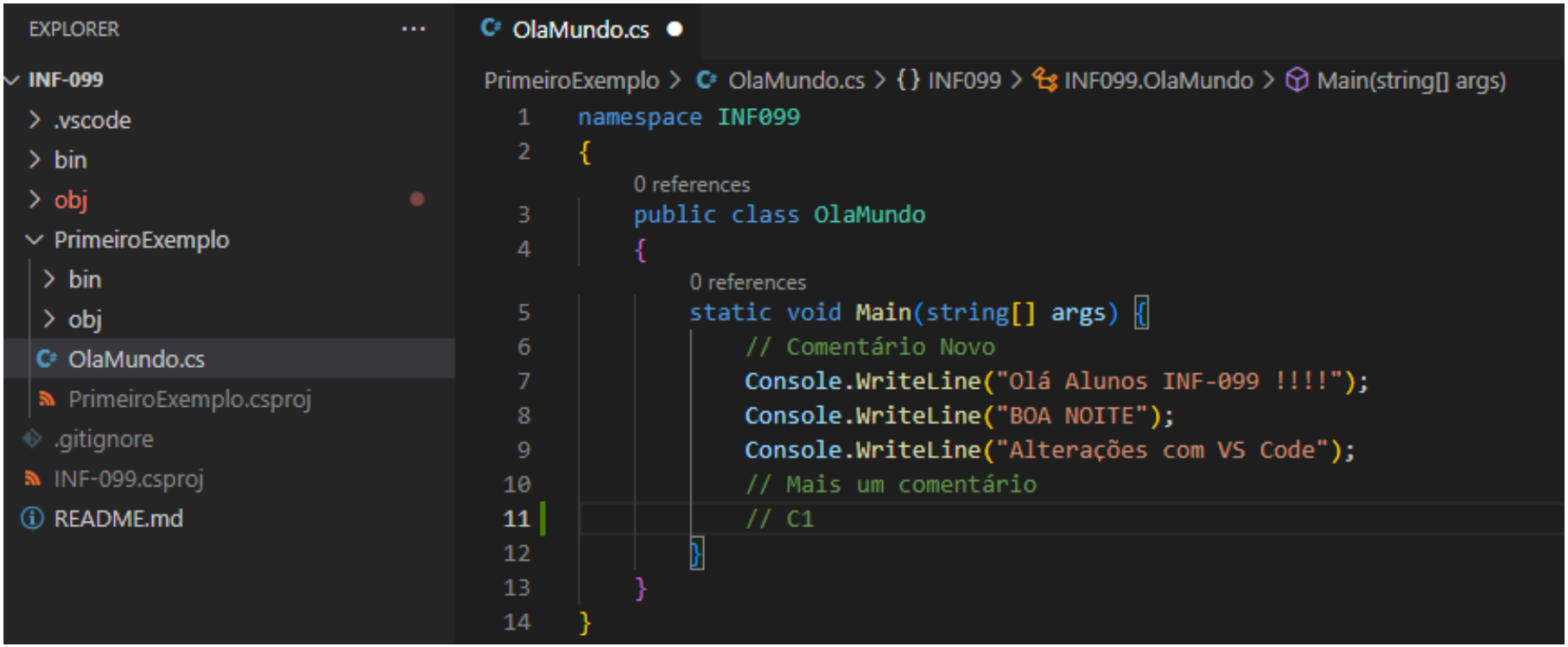
Prática com VS Code + GitHub

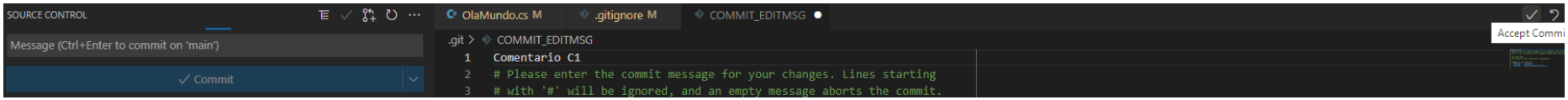
VS Code + GitHub

1) Vamos construir situação equivalente com VS Code + GitHub

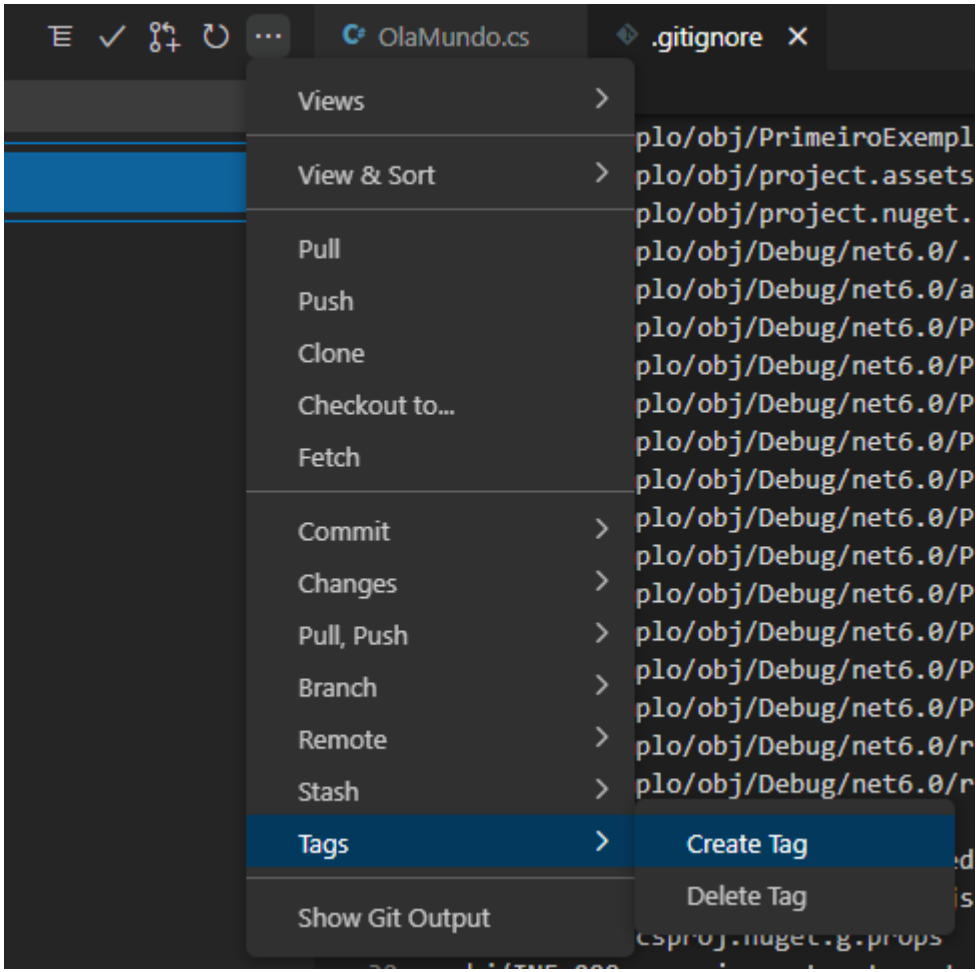


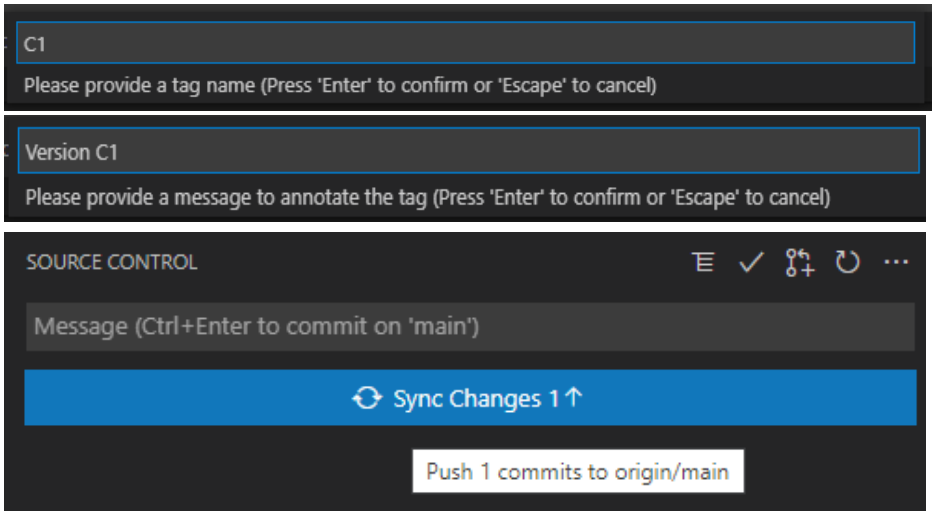
VS Code + GitHub

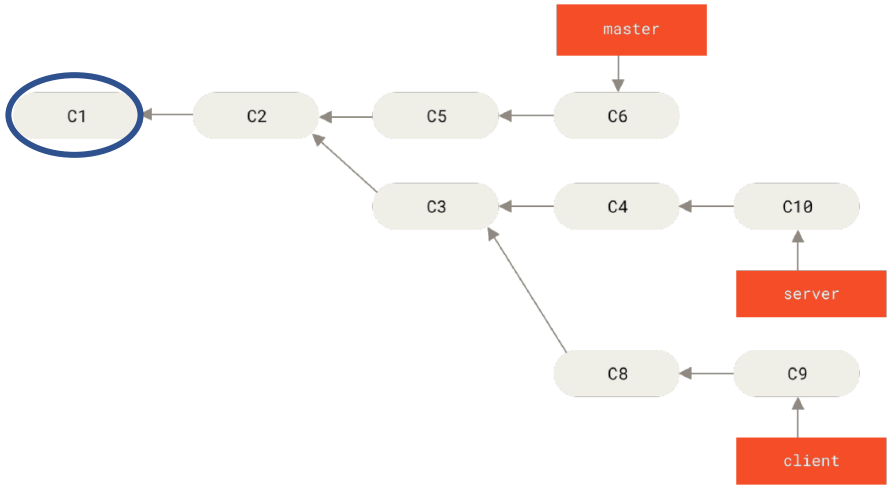




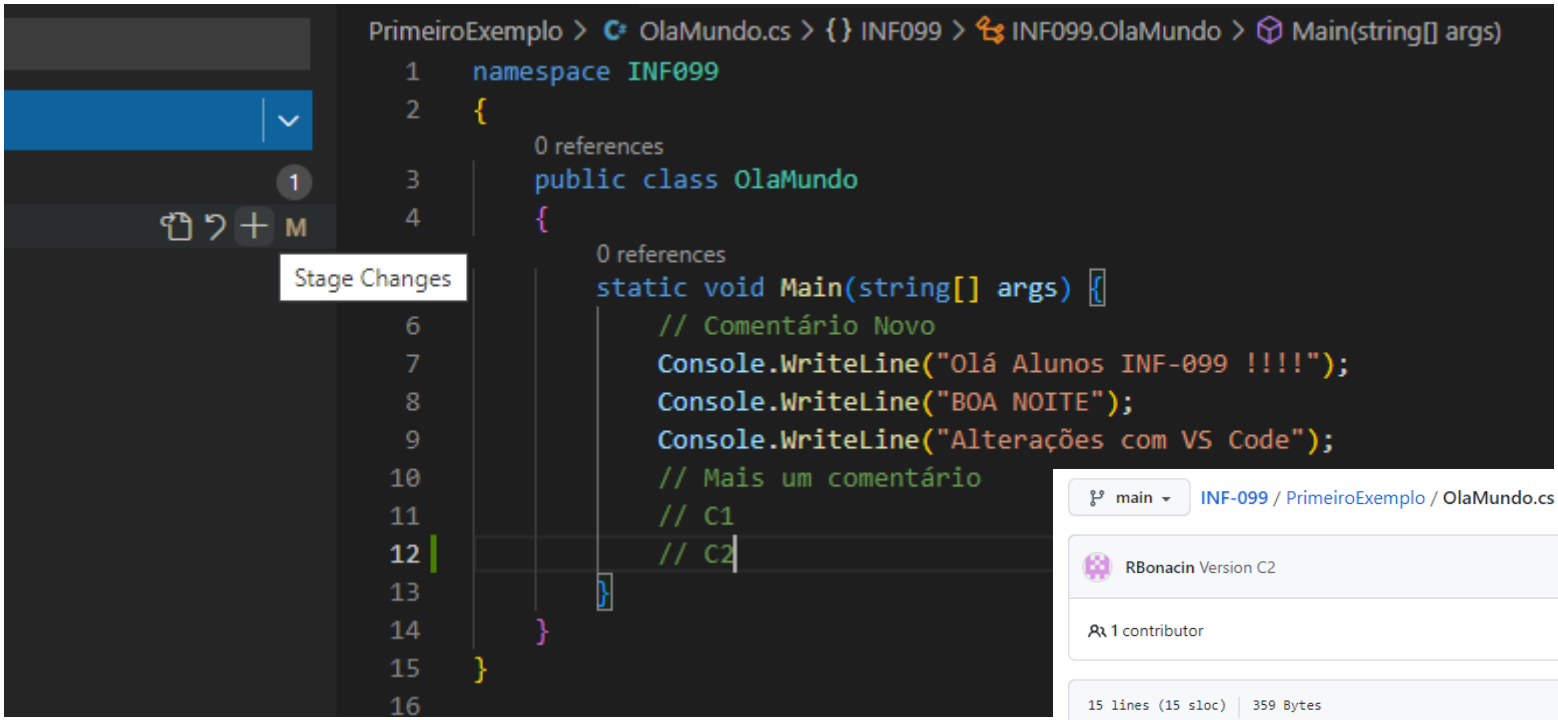
VS Code + GitHub

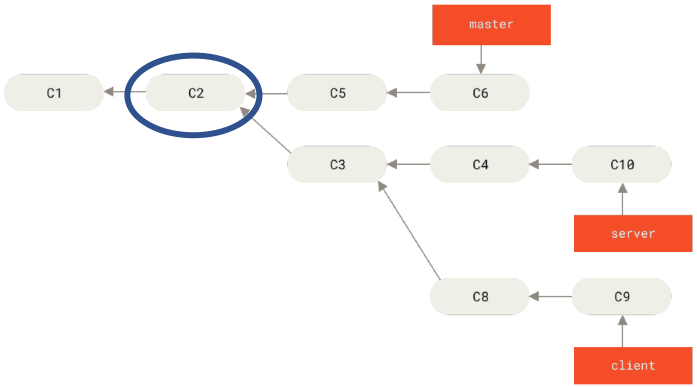




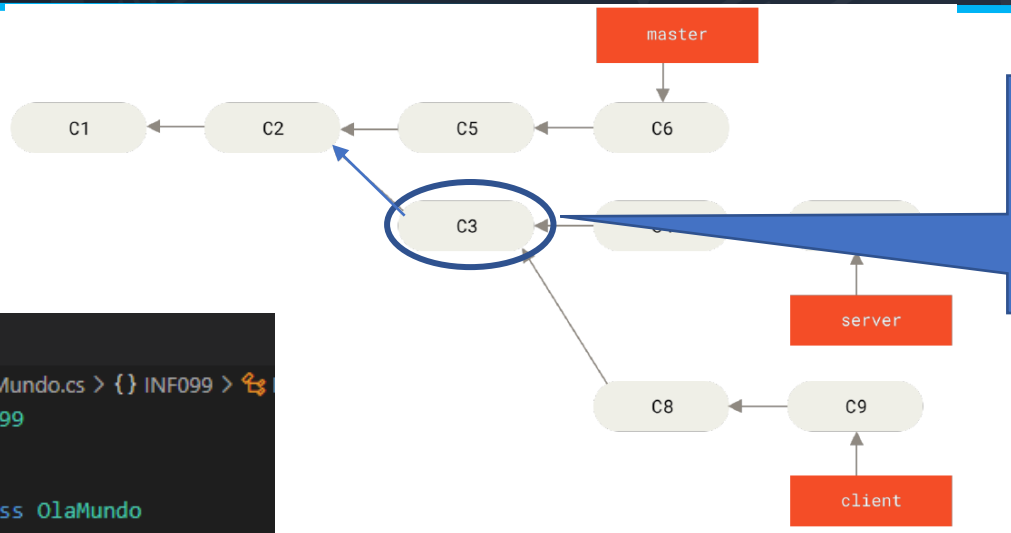


VS Code + GitHub

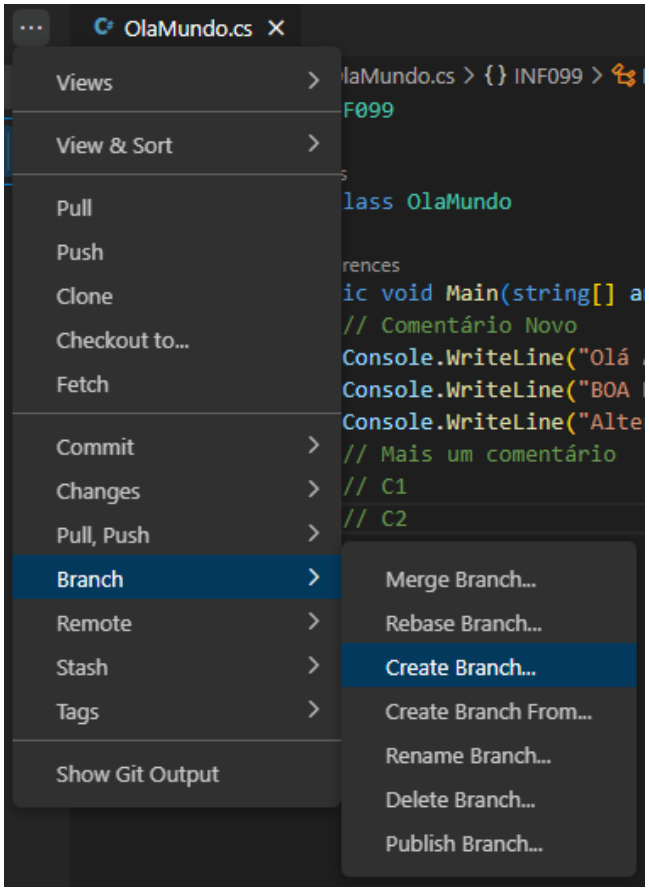


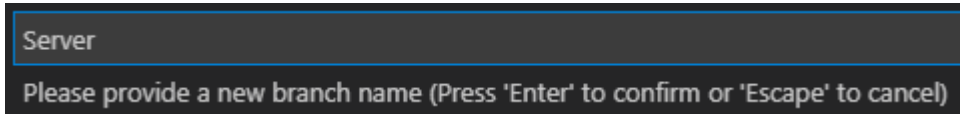


VS Code + GitHub

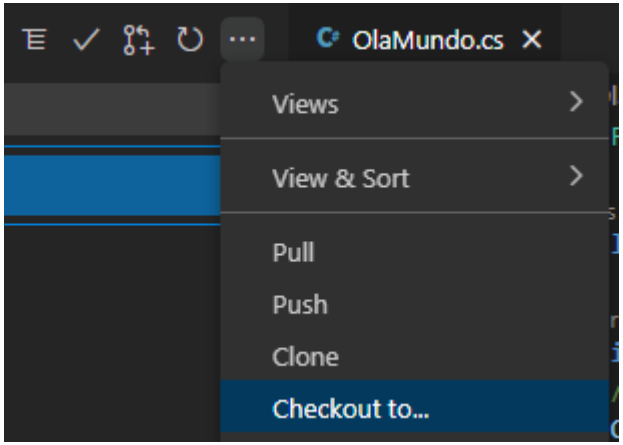


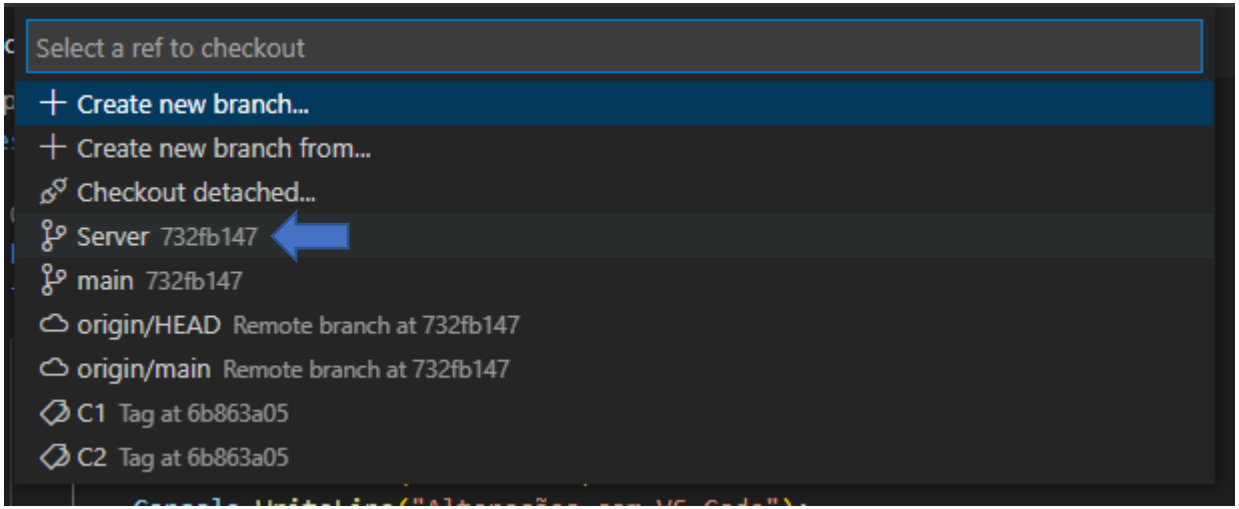
Criar Branch Server, fazer checkout, alterar arquivo, fazer commit e push

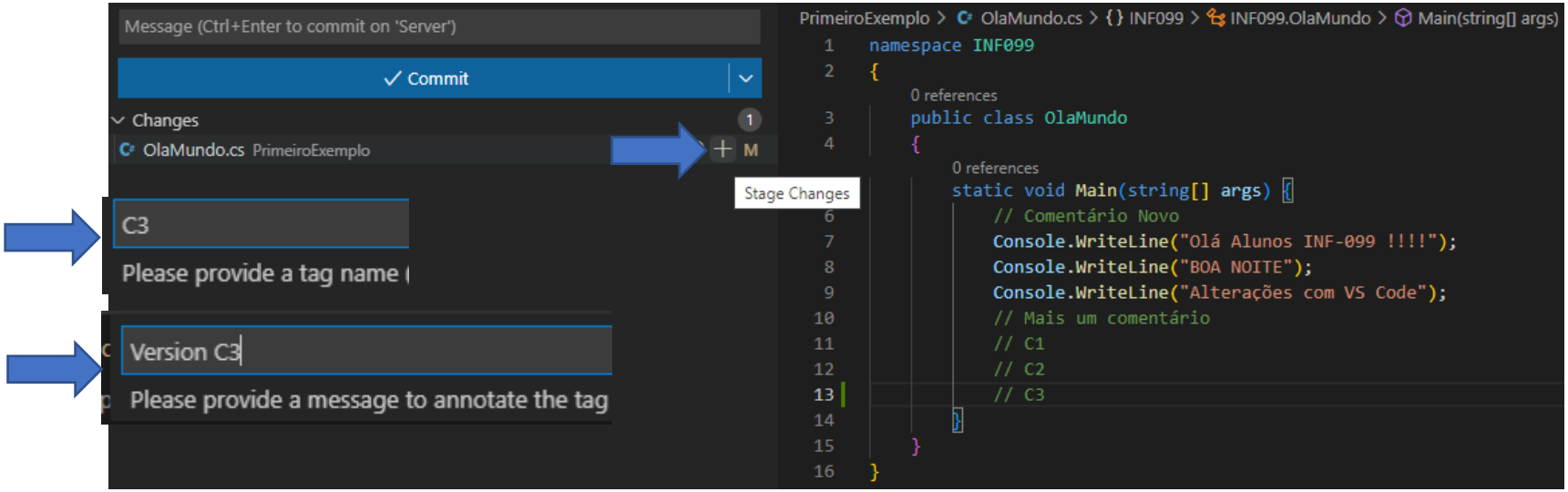




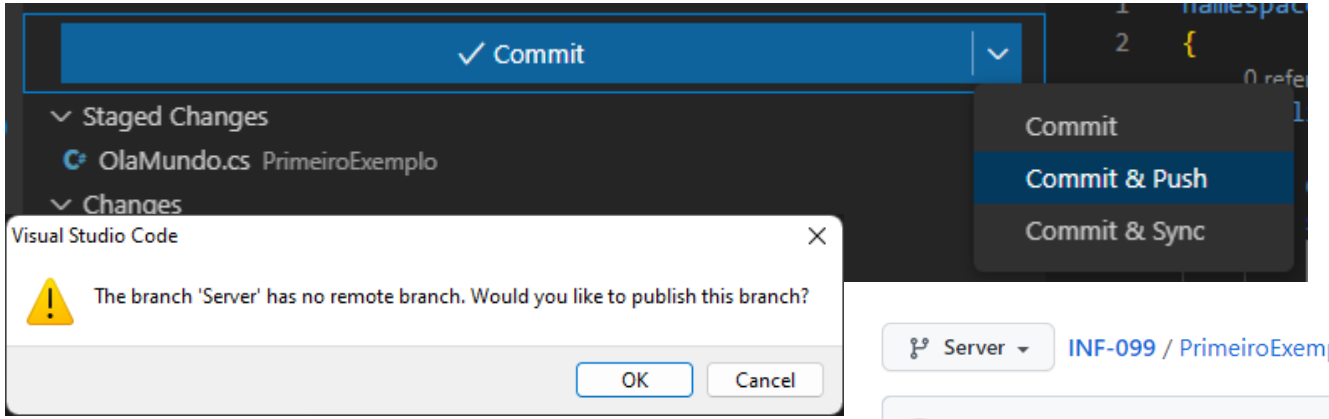
VS Code + GitHub

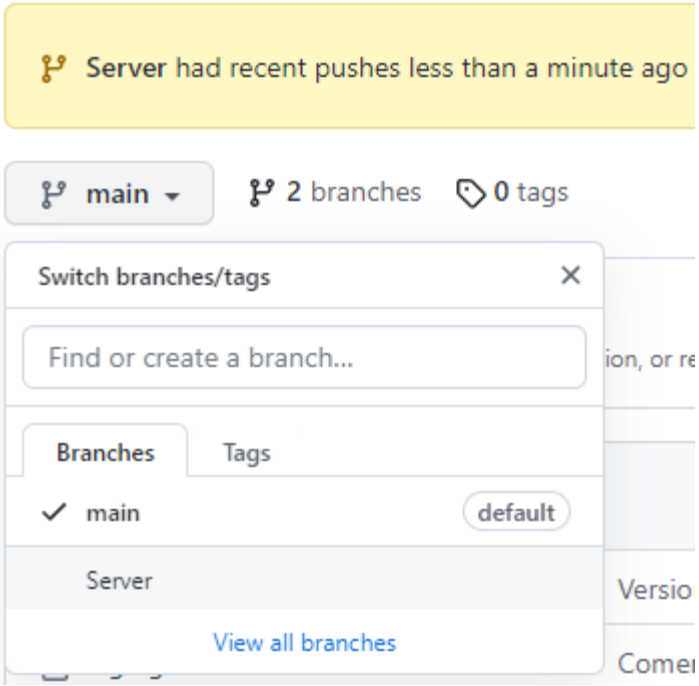




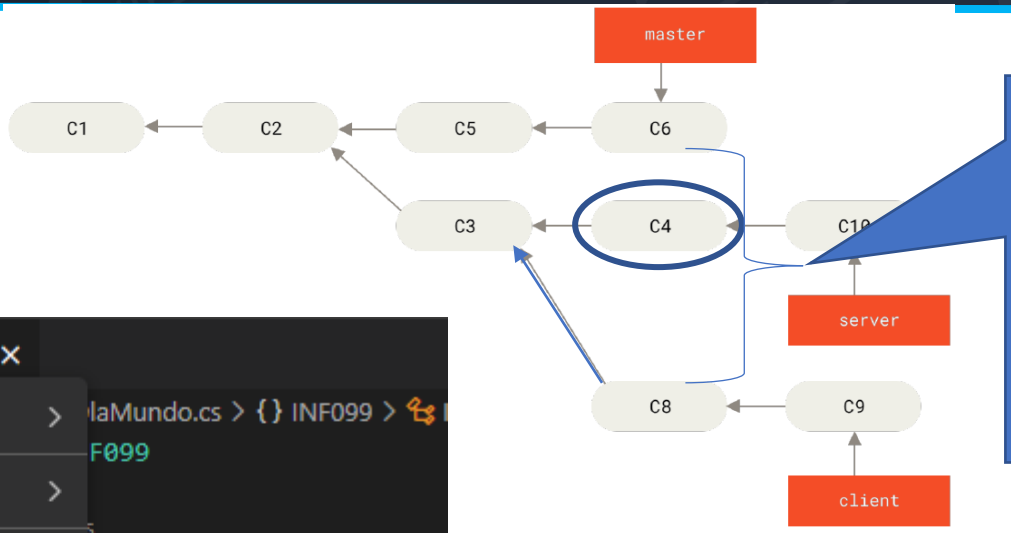


VS Code + GitHub

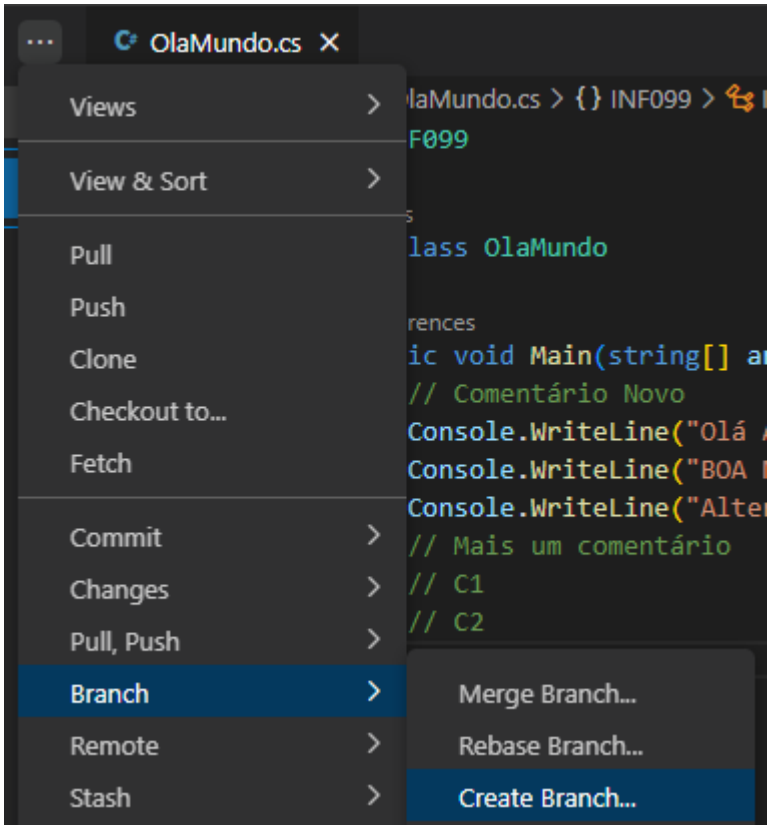


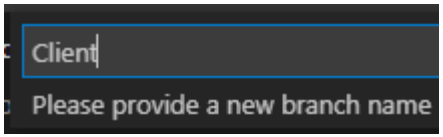


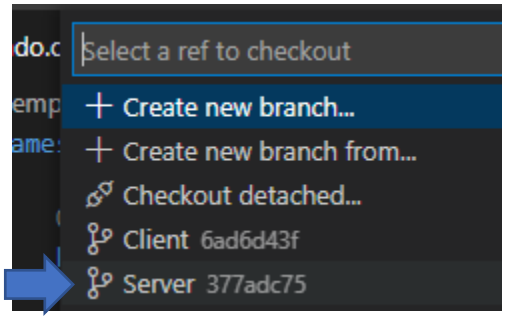
VS Code + GitHub



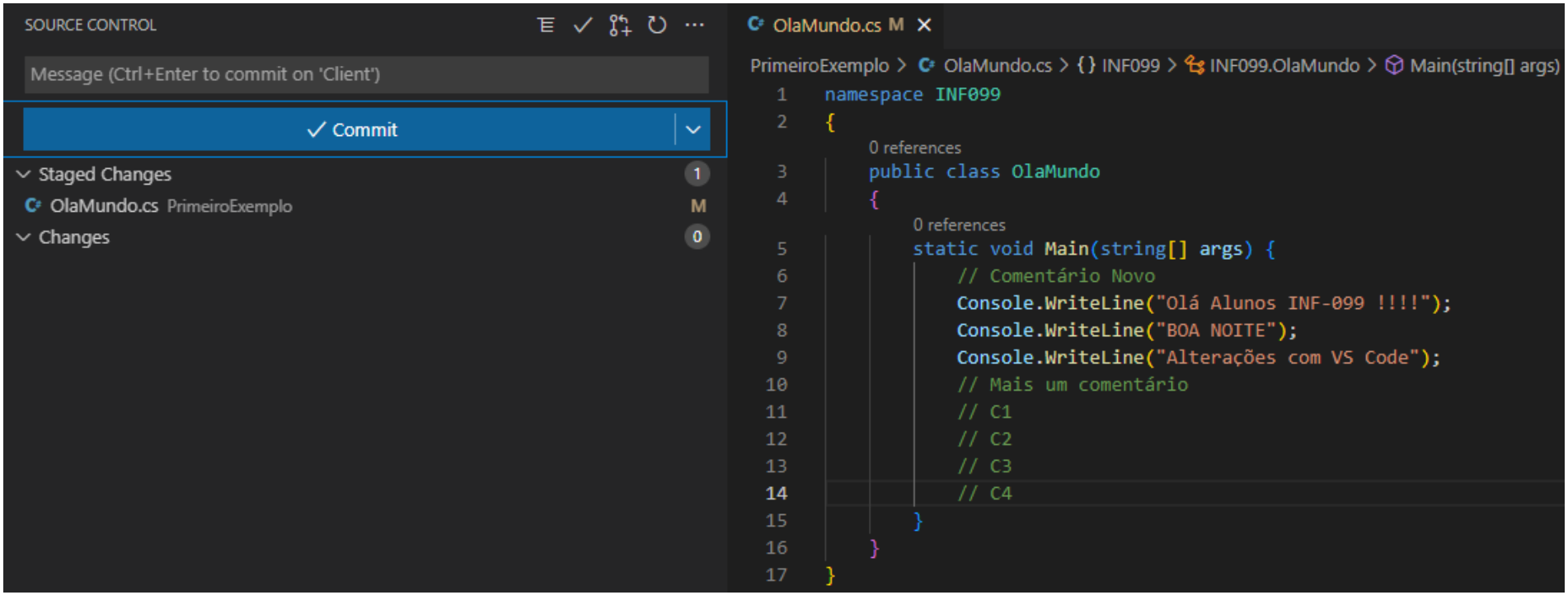
Criar Branch Client, mas fazer checkout “na Server”. Alterar arquivo, fazer commit e push “na server”

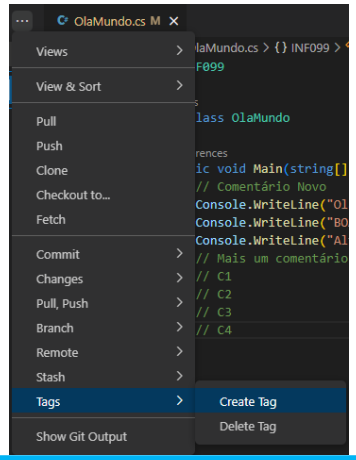


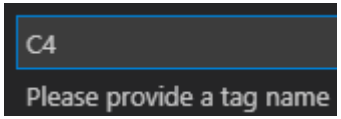


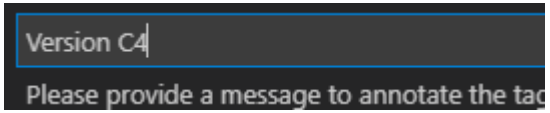


VS Code + GitHub

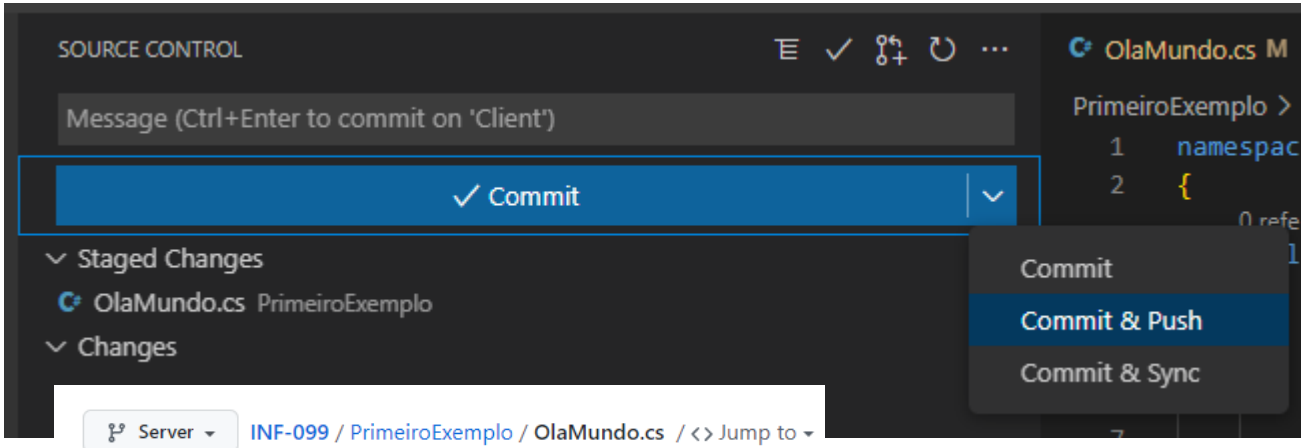








VS Code + GitHub



Server INF-099 / PrimeiroExemplo / OlaMundo.cs / Jump to

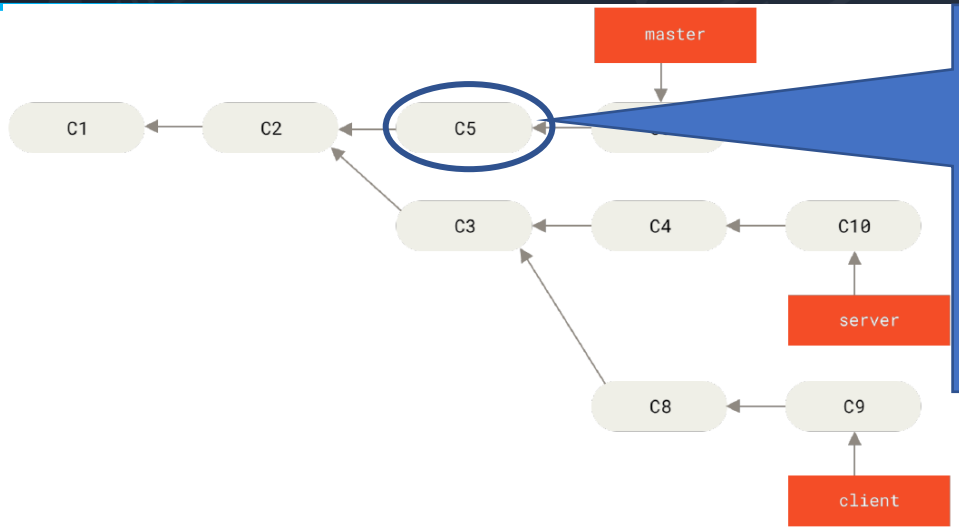
RBonacin Versão C4

1 contributor

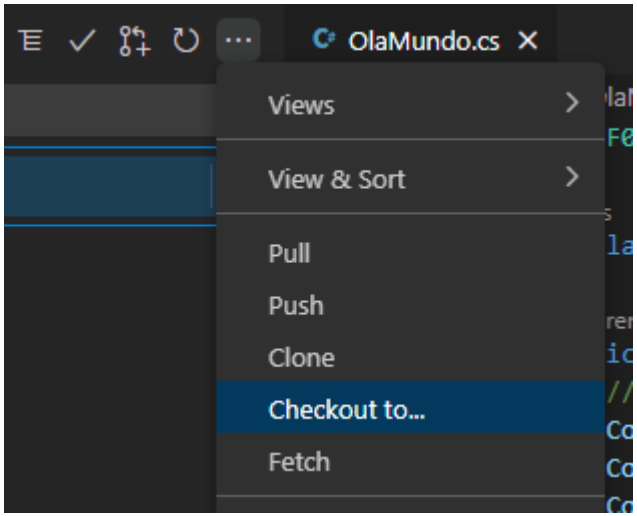
17 lines (17 sloc) | 395 Bytes

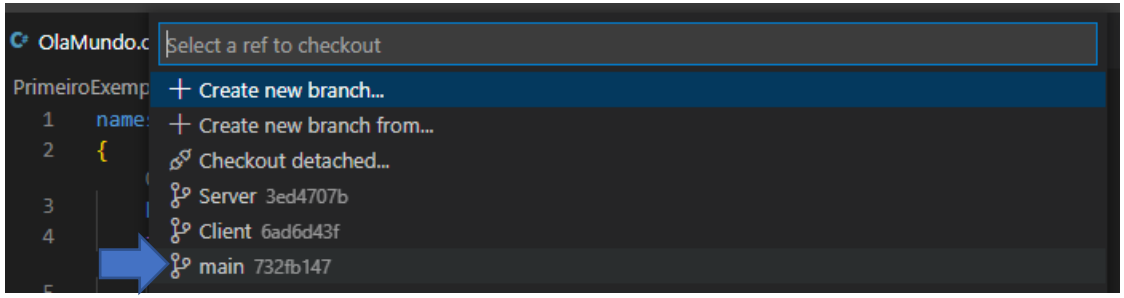
```
1  namespace INF099
2  {
3      public class OlaMundo
4      {
5          static void Main(string[] args) {
6                          // Comentário Novo
7              Console.WriteLine("Olá Alunos INF-099 !!!!");
8                          Console.WriteLine("BOA NOITE");
9              Console.WriteLine("Alterações com VS Code");
10             // Mais um comentário
11             // C1
12             // C2
13             // C3
14             // C4
15         }
16     }
17 }
```

VS Code + GitHub

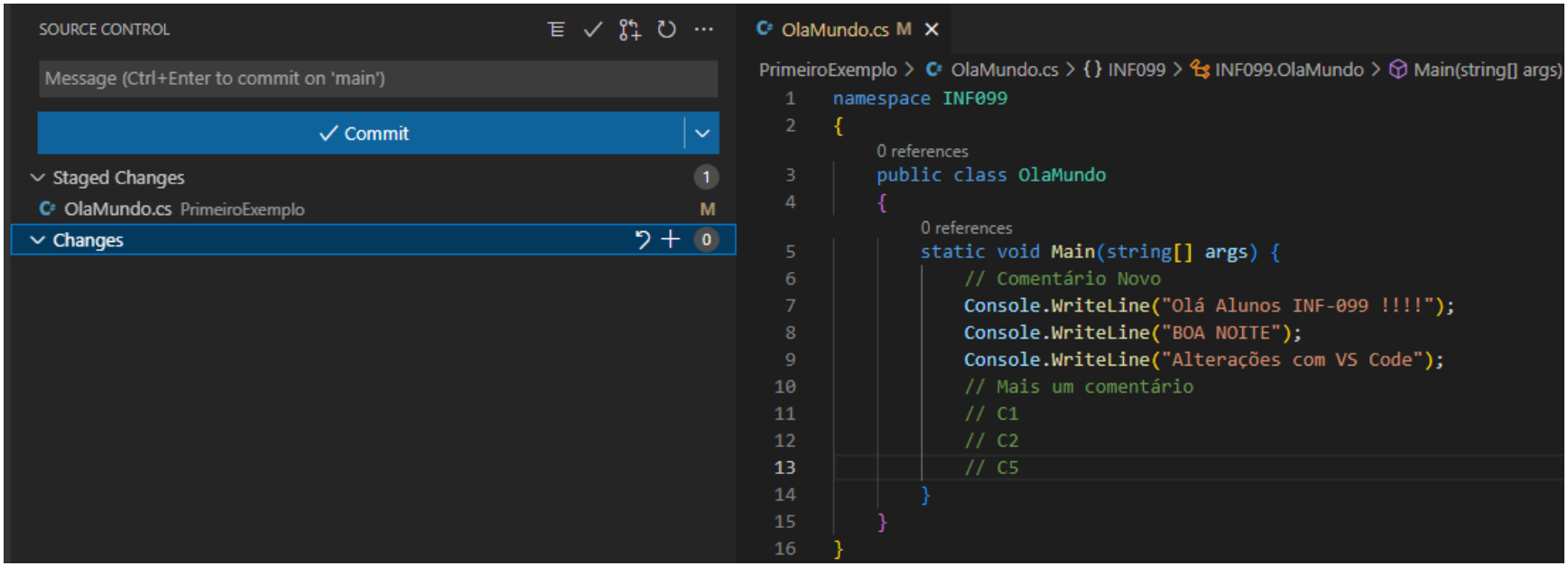


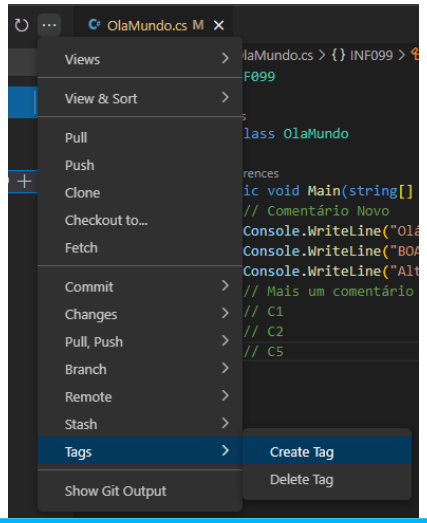
Fazer checkout “na Main/Master”. Alterar arquivo, fazer commit e push

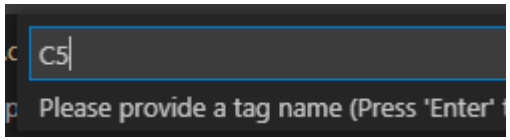


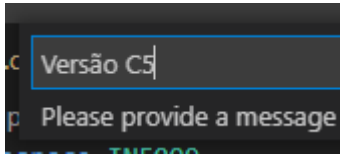


VS Code + GitHub

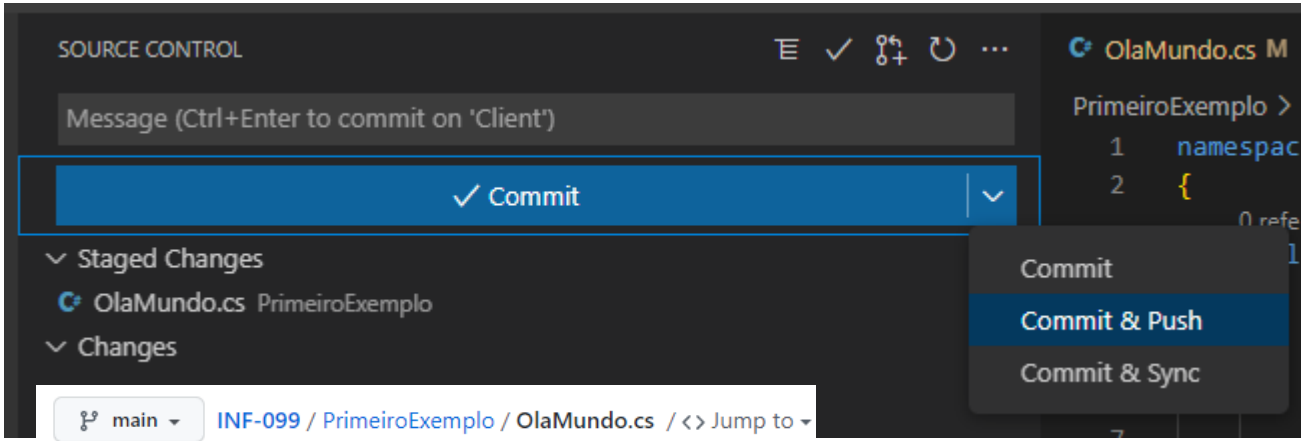








VS Code + GitHub



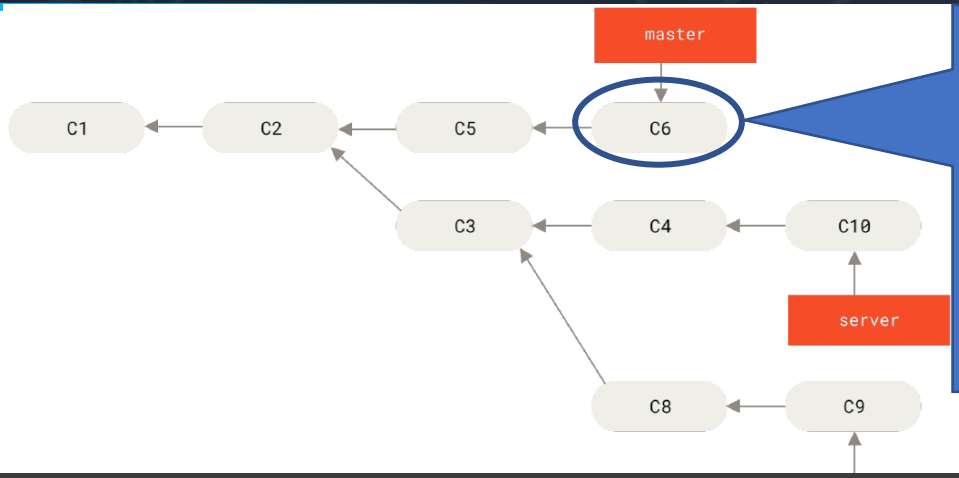
RBonacin Version C5

1 contributor

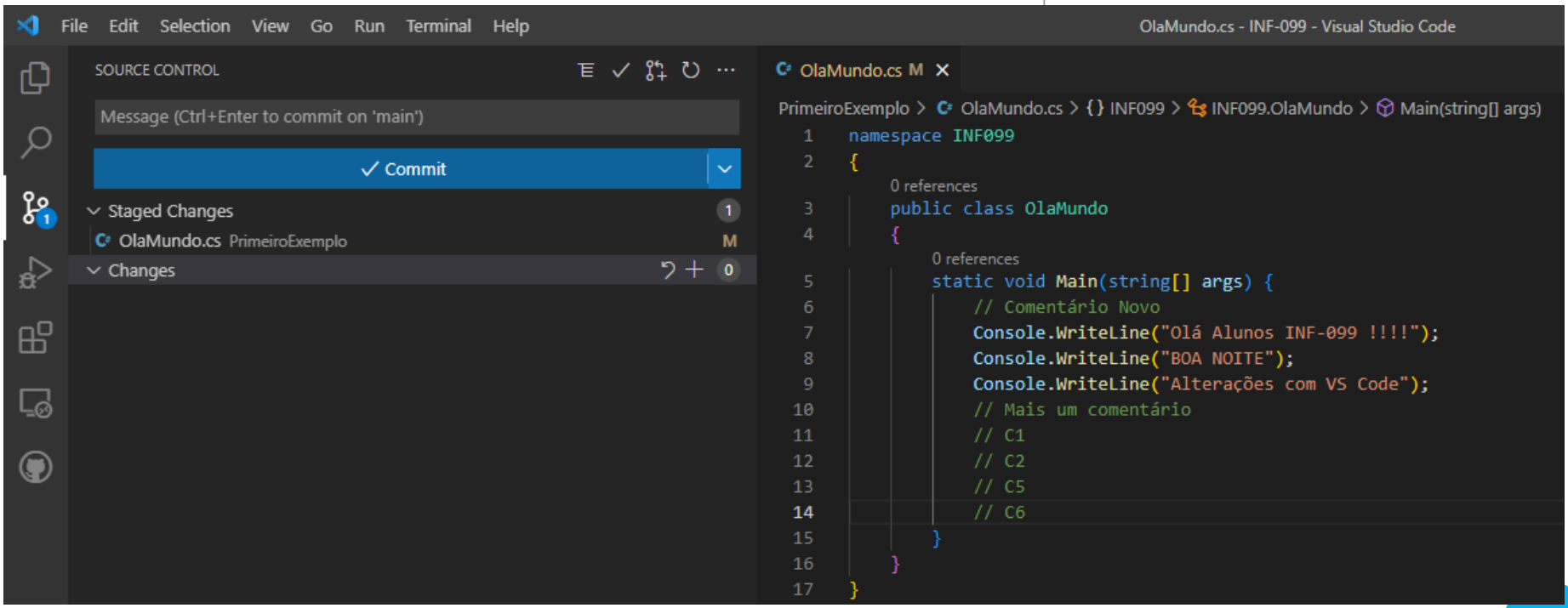
16 lines (16 sloc) | 377 Bytes

```
1  namespace INF099
2  {
3      public class OlaMundo
4      {
5          static void Main(string[] args) {
6                          // Comentário Novo
7              Console.WriteLine("Olá Alunos INF-099 !!!!");
8                          Console.WriteLine("BOA NOITE");
9              Console.WriteLine("Alterações com VS Code");
10             // Mais um comentário
11             // C1
12             // C2
13             // C5
14         }
15     }
16 }
```

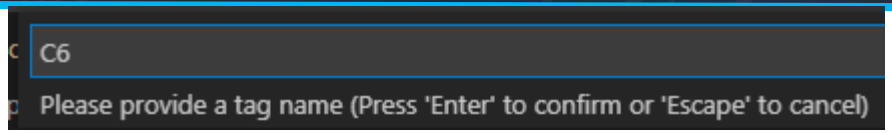
VS Code + GitHub

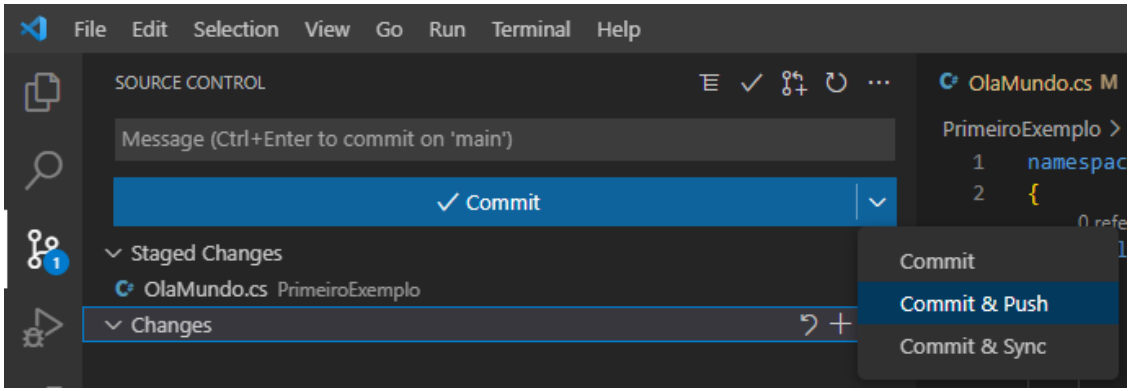


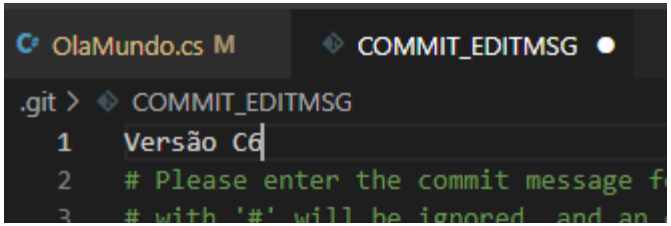
Continuar “na Main/Master”. Alterar arquivo, fazer commit e push



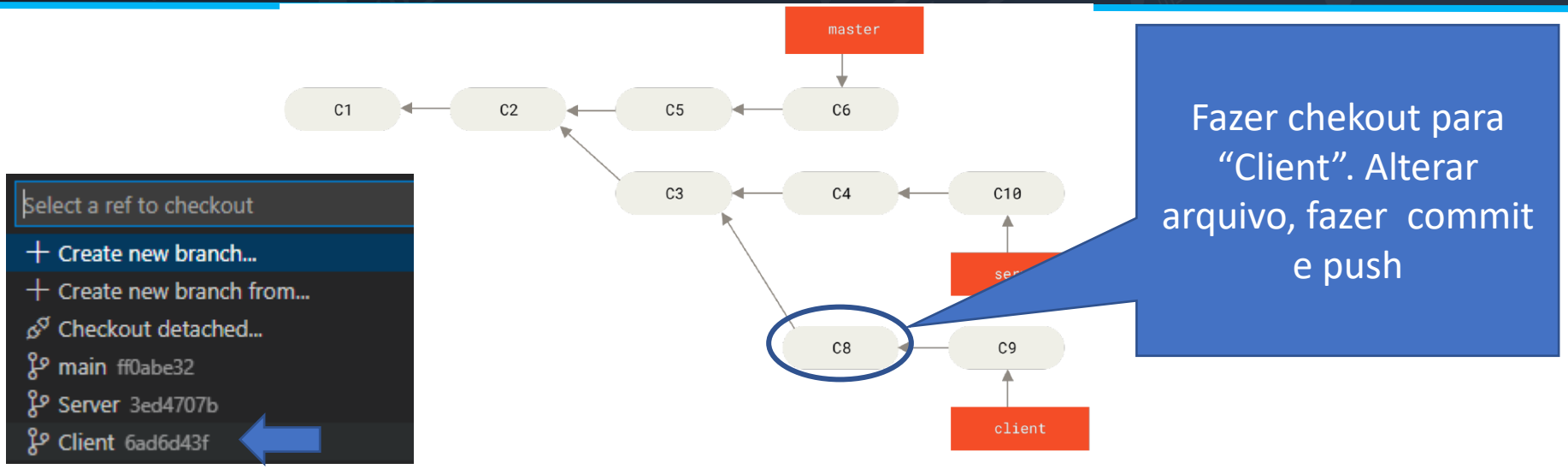
VS Code + GitHub

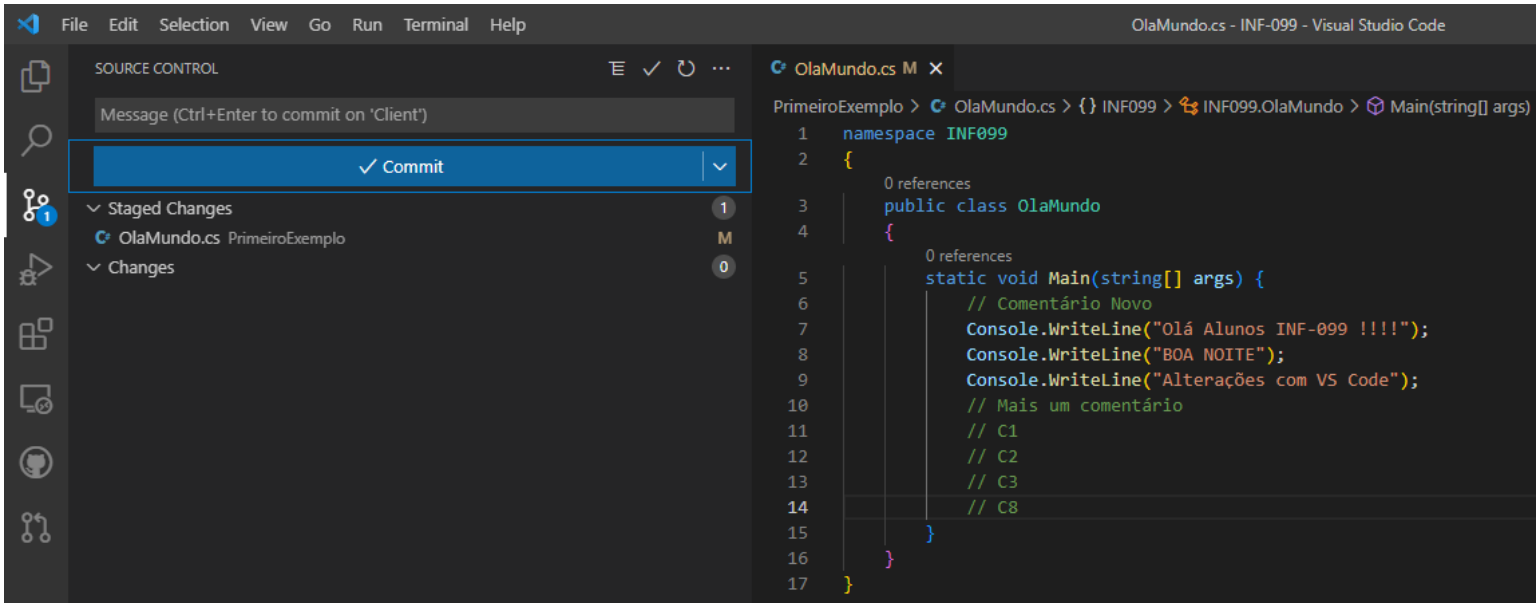




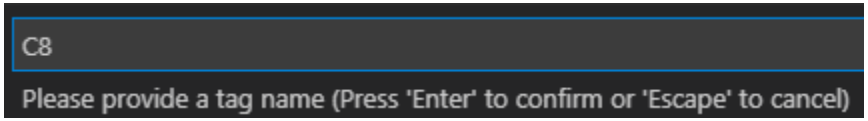


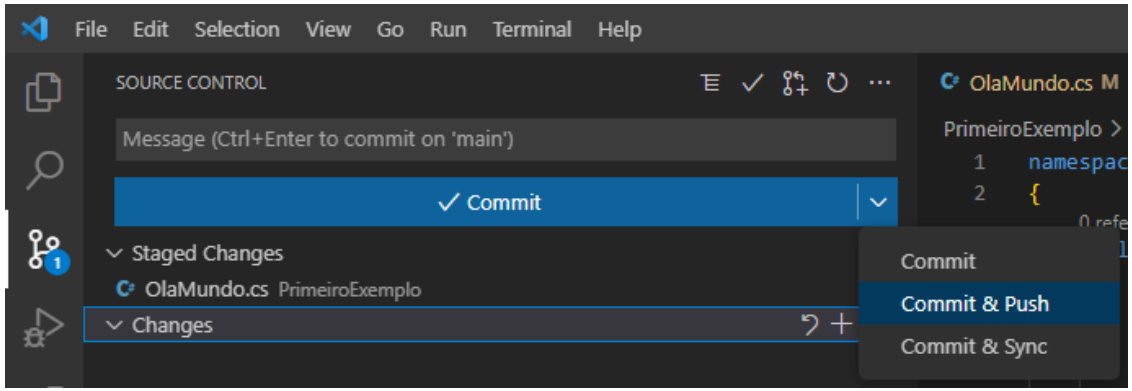
VS Code + GitHub

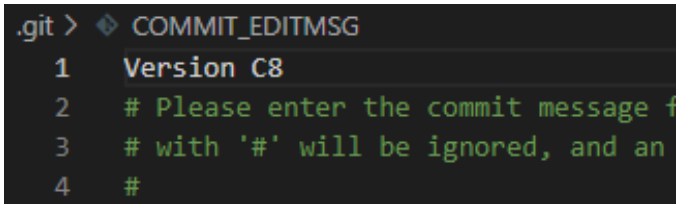




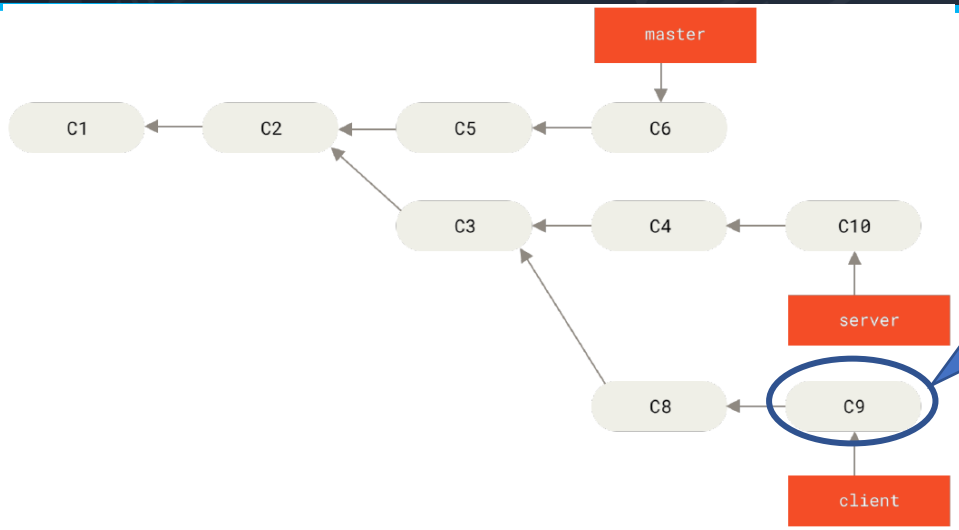
VS Code + GitHub



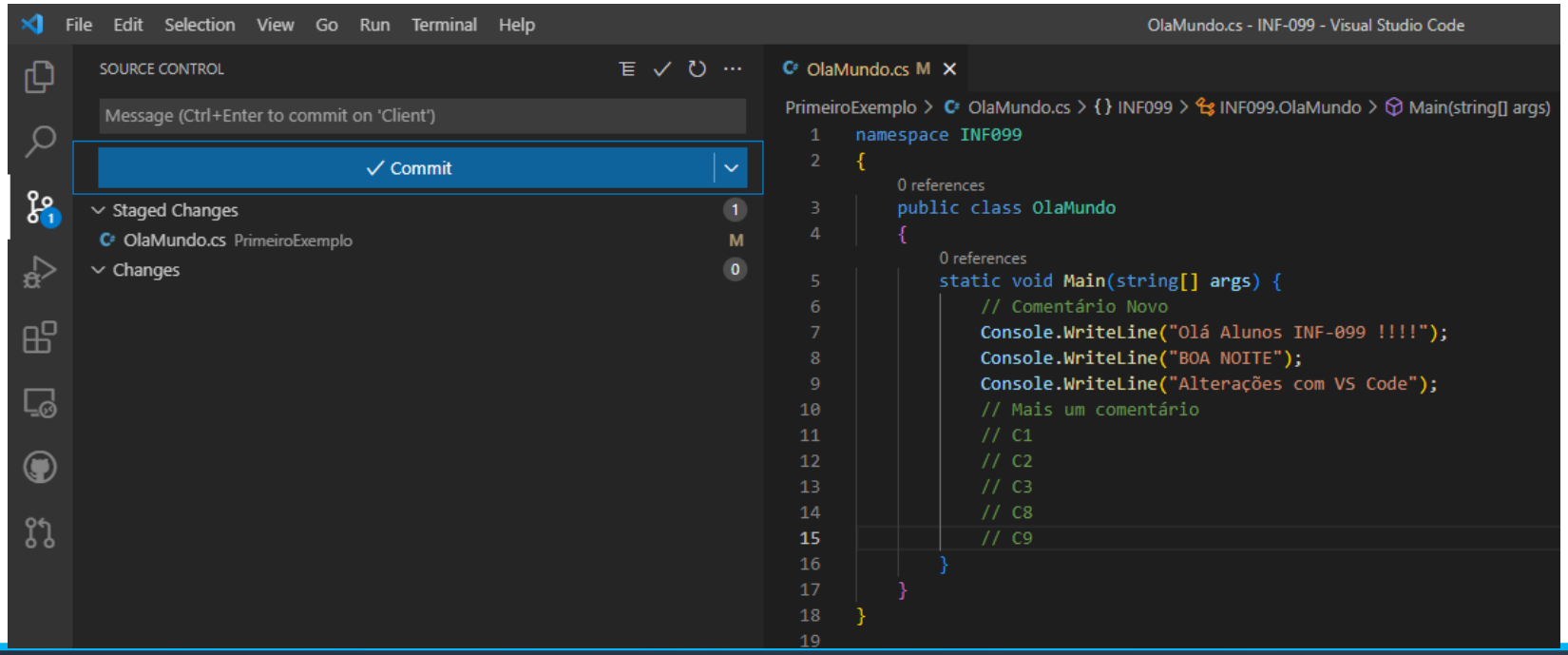




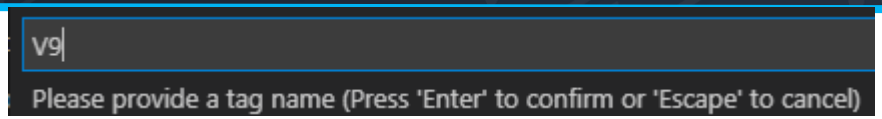
VS Code + GitHub

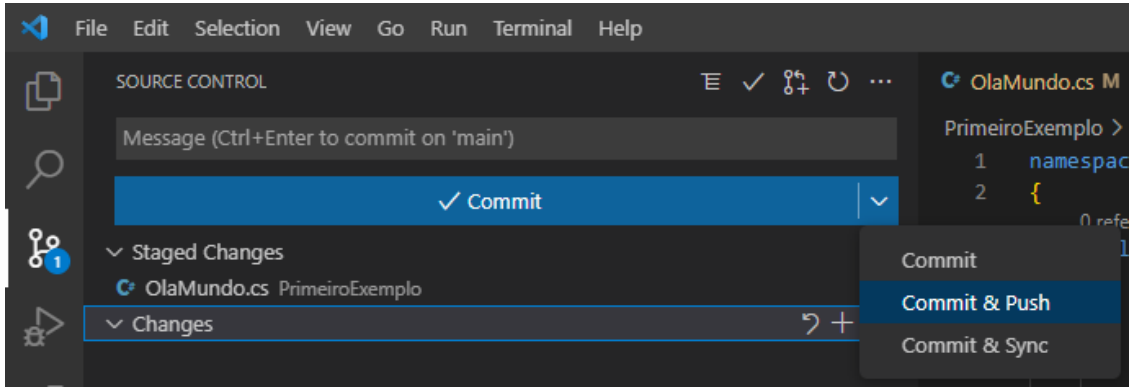


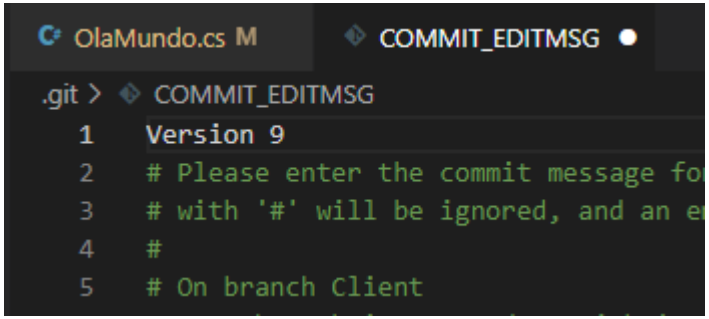
Continuar na “Client”. Alterar arquivo, fazer commit e push



VS Code + GitHub







Client INF-099 / PrimeiroExemplo / OlaMundo.cs / Jump to

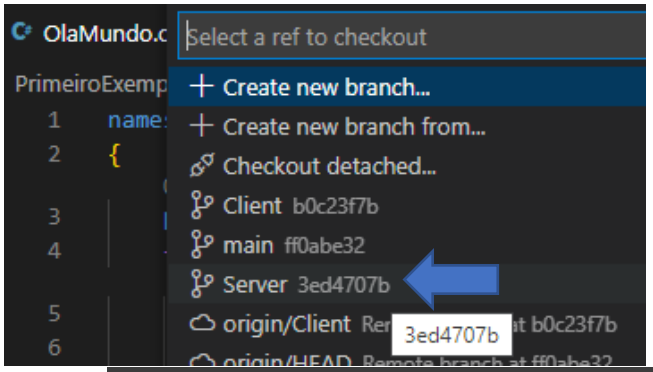
RBonacin Version 9

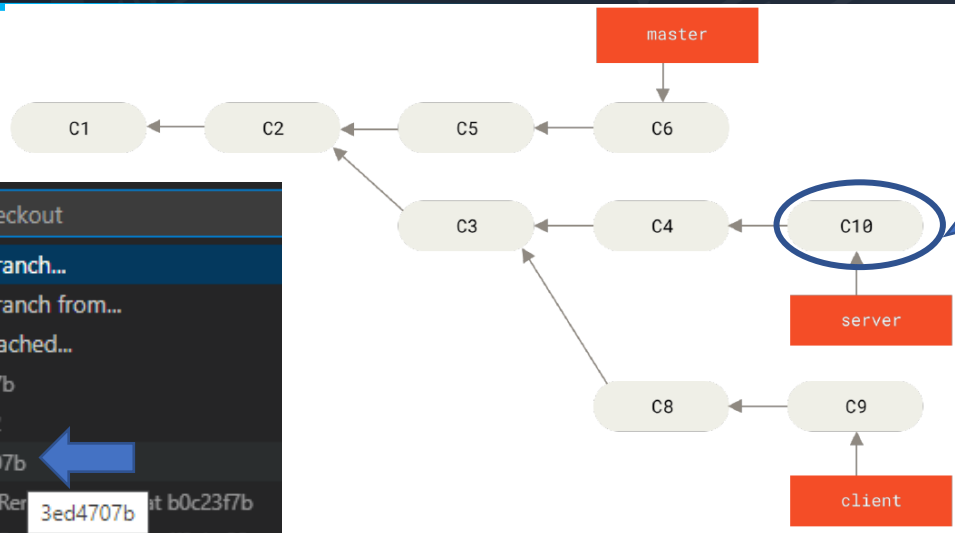
1 contributor

18 lines (18 sloc) | 413 Bytes

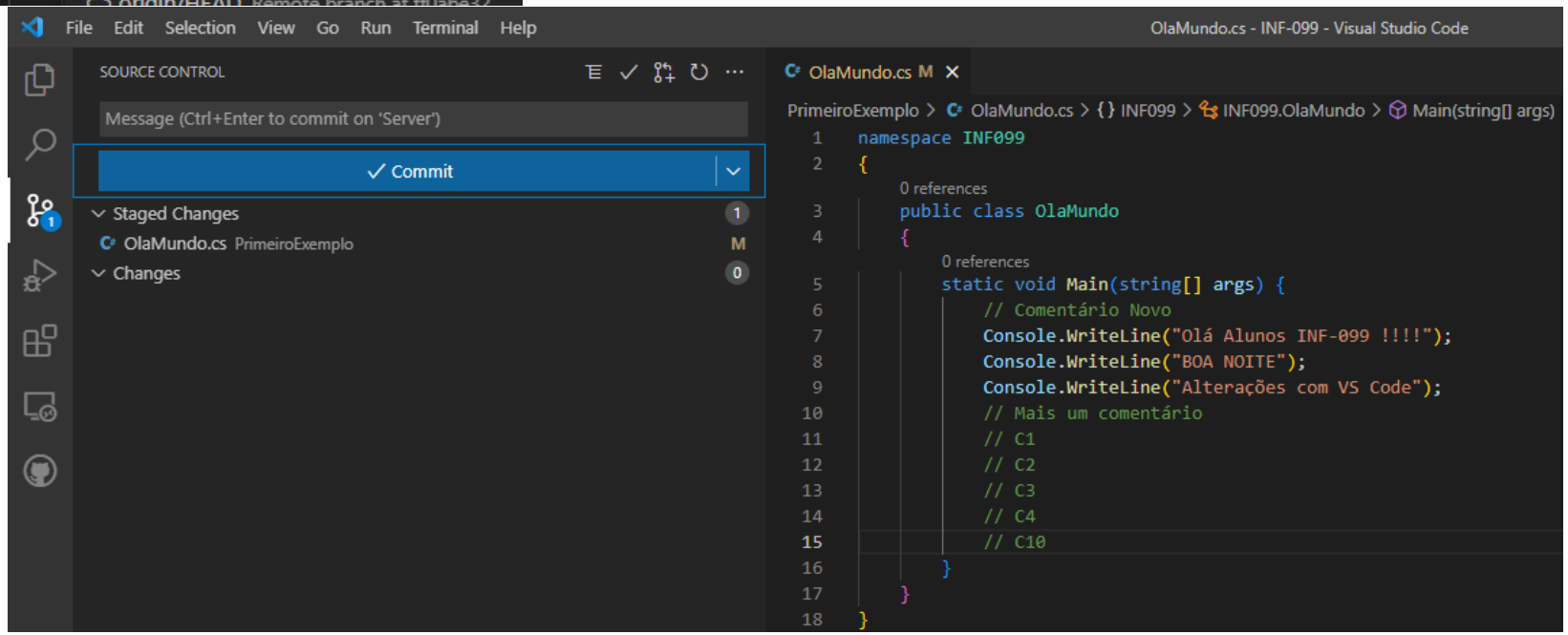
```
1  namespace INF099
2  {
3      public class OlaMundo
4      {
5          static void Main(string[] args) {
6                          // Comentário Novo
7              Console.WriteLine("Olá Alunos INF-099 !!!!");
8                          Console.WriteLine("BOA NOITE");
9              Console.WriteLine("Alterações com VS Code");
10             // Mais um comentário
11             // C1
12             // C2
13             // C3
14             // C8
15             // C9
16         }
17     }
18 }
```

VS Code + GitHub

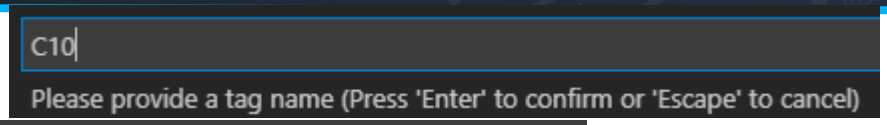


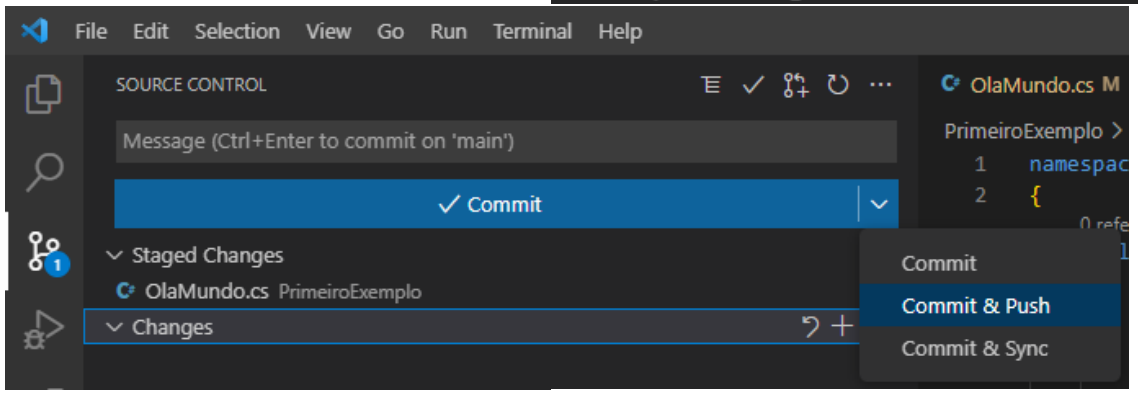


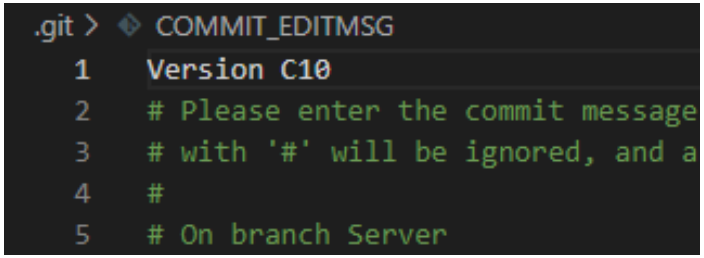
Fazer checkout na "Server". Alterar arquivo, fazer commit e push



VS Code + GitHub







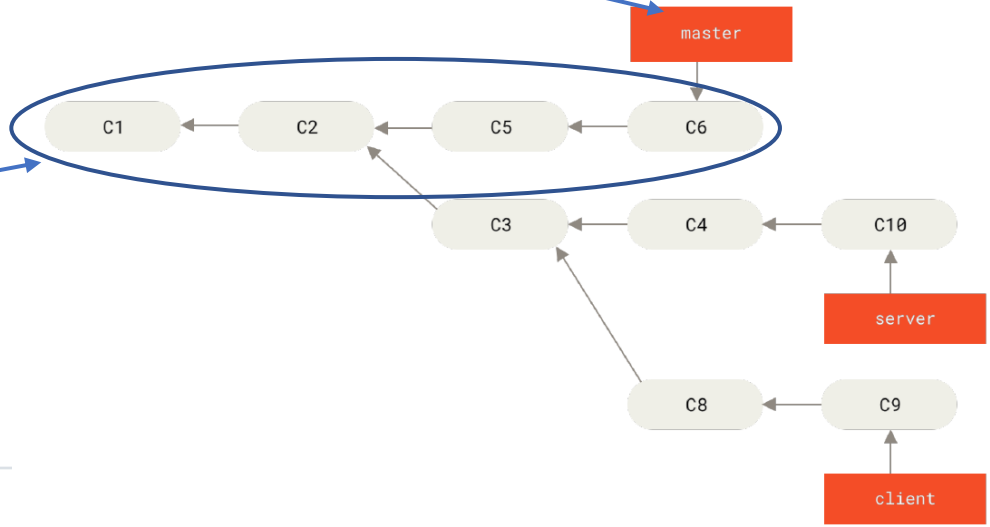
VS Code + GitHub

2) Conferir se está correto ...

main

5 Bytes

```
1   namespace INF099
2   {
3       public class OlaMundo
4       {
5           static void Main(string[] args) {
6                           // Comentário Novo
7               Console.WriteLine("Olá Alunos INF-099 !!!!");
8                           Console.WriteLine("BOA NOITE");
9               Console.WriteLine("Alterações com VS Code");
10              // Mais um comentário
11              // C1
12              // C2
13              // C5
14              // C6
15          }
16      }
17  }
```

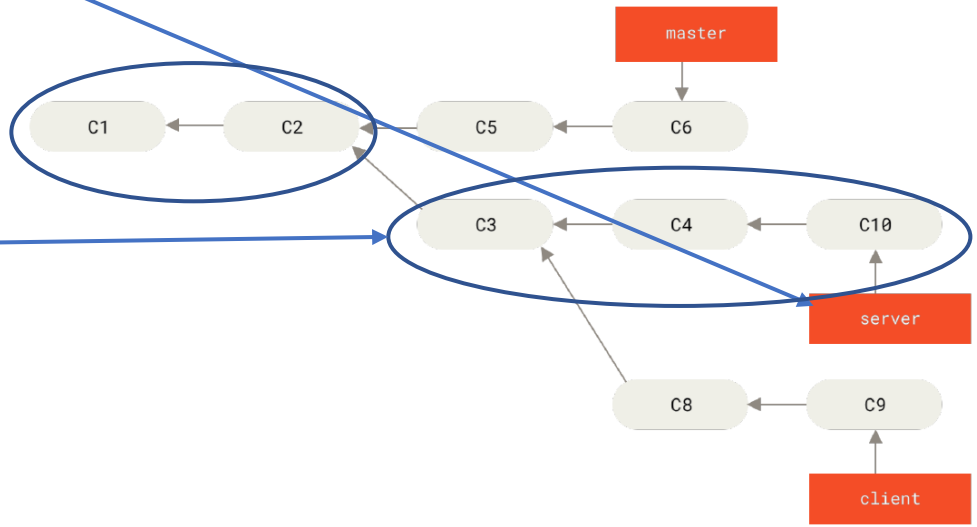


VS Code + GitHub

2) Conferir se está correto ...

Server | 414 Bytes

```
1   namespace INF099
2   {
3       public class OlaMundo
4       {
5           static void Main(string[] args) {
6                           // Comentário Novo
7               Console.WriteLine("Olá Alunos INF-099 !!!!");
8                           Console.WriteLine("BOA NOITE");
9               Console.WriteLine("Alterações com VS Code");
10              // Mais um comentário
11              // C1
12              // C2
13              // C3
14              // C4
15              // C10
16          }
17      }
18  }
```

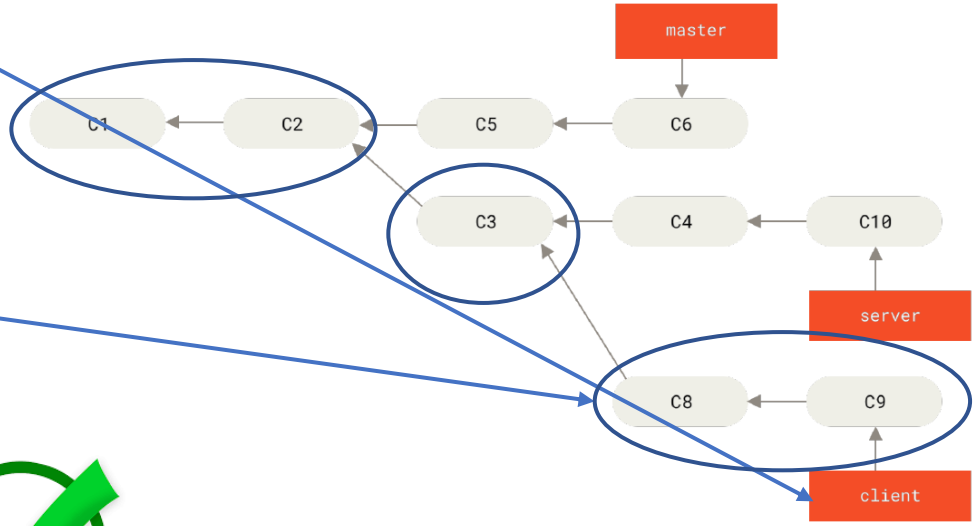


VS Code + GitHub

2) Conferir se está correto ...

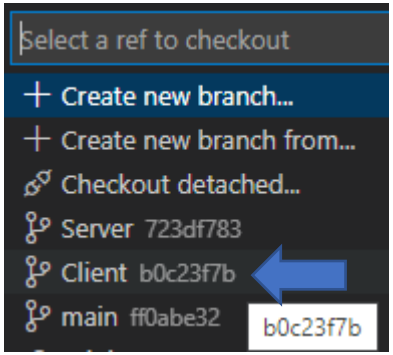
Client | 413 Bytes

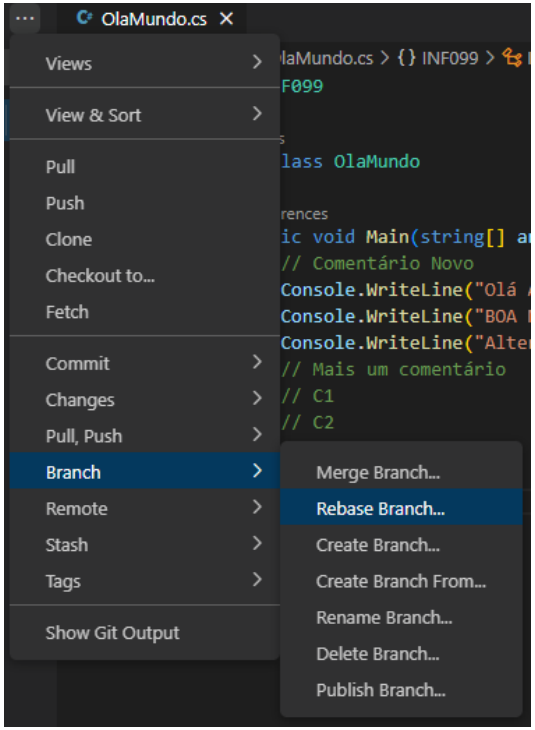
```
1  namespace INF099
2  {
3      public class OlaMundo
4      {
5          static void Main(string[] args) {
6                          // Comentário Novo
7              Console.WriteLine("Olá Alunos INF-099 !!!!");
8                          Console.WriteLine("BOA NOITE");
9              Console.WriteLine("Alterações com VS Code");
10             // Mais um comentário
11             // C1
12             // C2
13             // C3
14             // C8
15             // C9
16         }
17     }
18 }
```

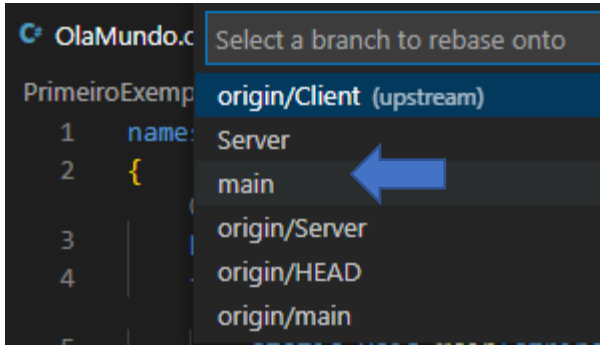


VS Code + GitHub

3) Fazer *rebase* da “client” na “master” (*main*)

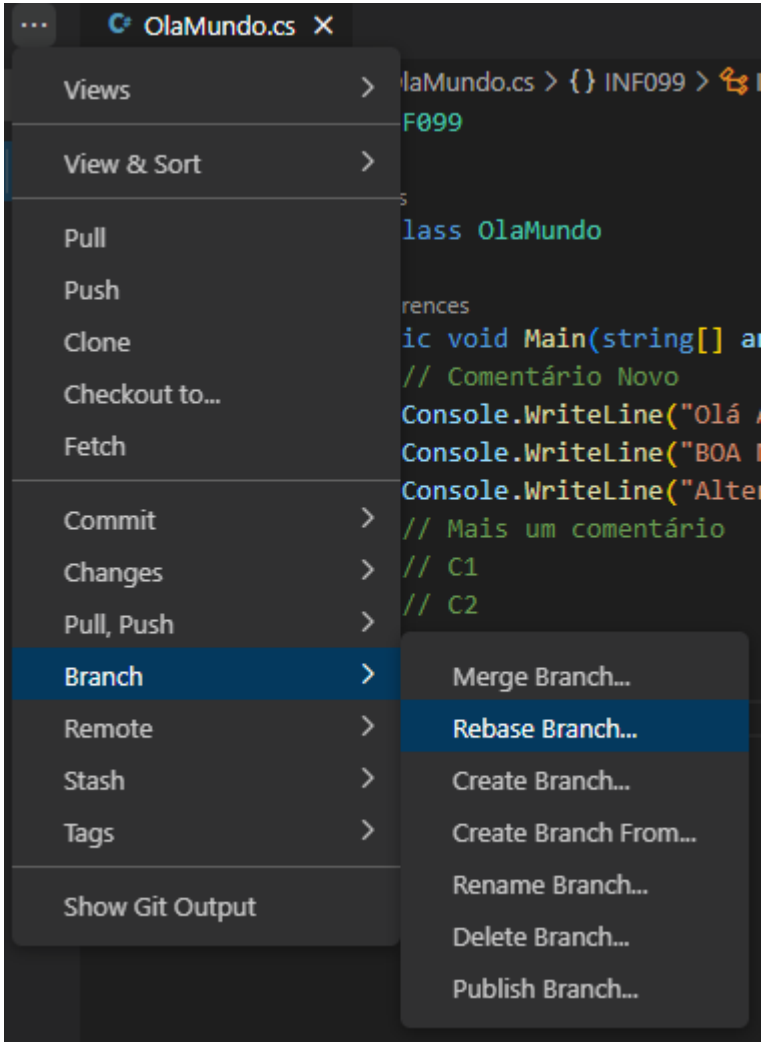


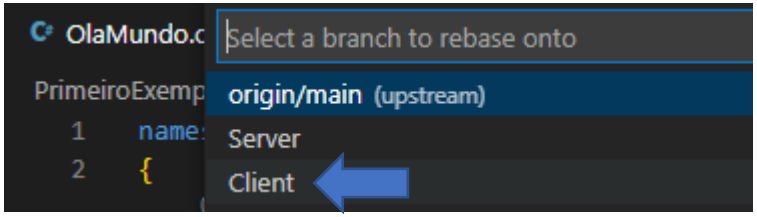




VS Code + GitHub

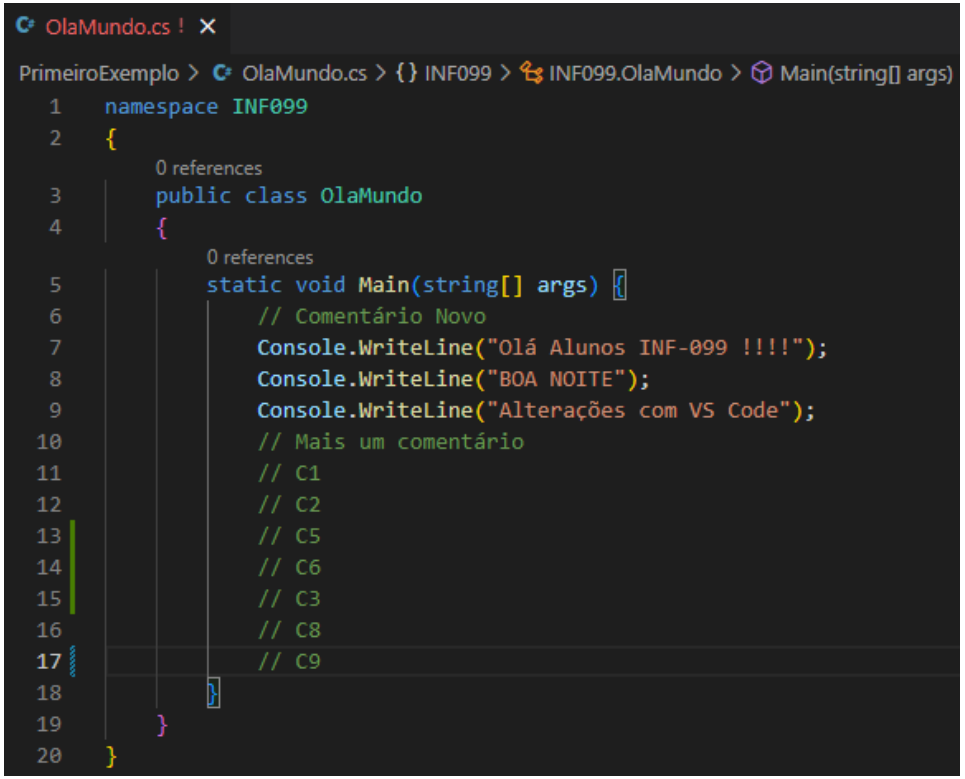
3) Fazer *rebase* da “client” na “master” (*main*)

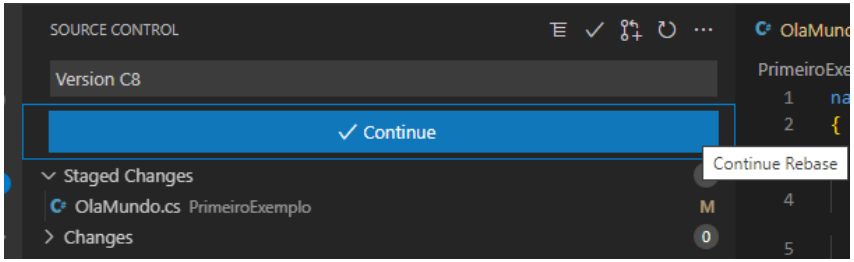


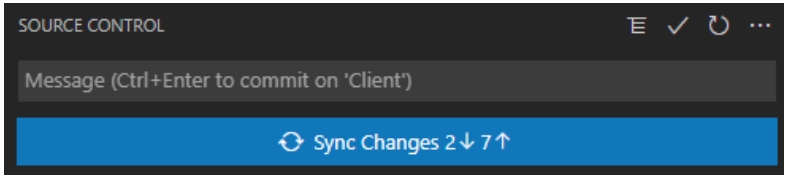


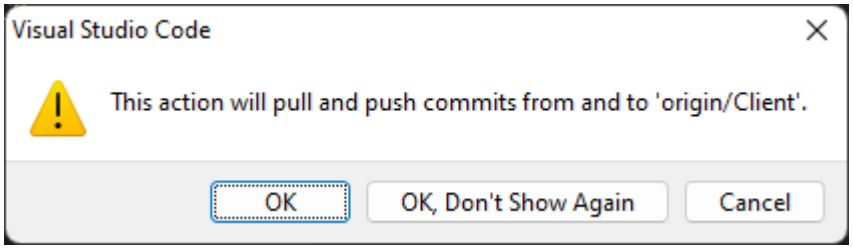
VS Code + GitHub

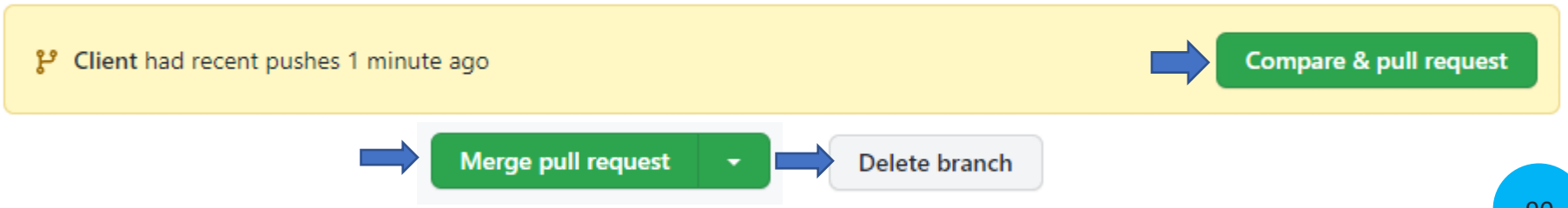
3) Fazer *rebase* da “client” na “master” (*main*)











VS Code + GitHub

3) Fazer *rebase* da “client” na “master” (*main*)

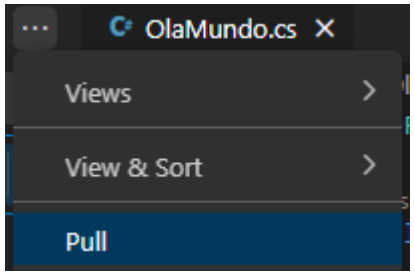
main ▾ INF-099 / PrimeiroExemplo / OlaMundo.cs / <> Jump to ▾

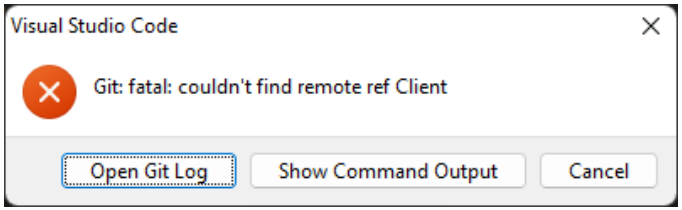
RBonacin Version C8

1 contributor

20 lines (20 sloc) | 449 Bytes

```
1   namespace INF099
2   {
3       public class OlaMundo
4       {
5           static void Main(string[] args) {
6                           // Comentário Novo
7               Console.WriteLine("Olá Alunos INF-099 !!!!");
8                           Console.WriteLine("BOA NOITE");
9               Console.WriteLine("Alterações com VS Code");
10              // Mais um comentário
11              // C1
12              // C2
13              // C5
14              // C6
15              // C3
16              // C8
17              // C9
18          }
19      }
20  }
```

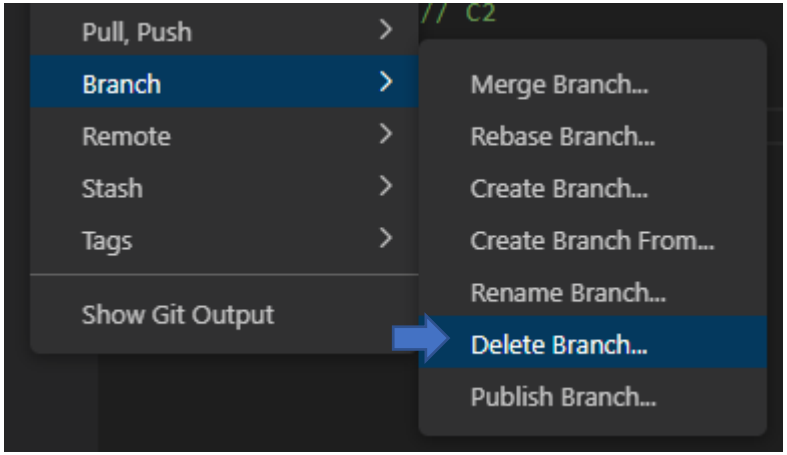


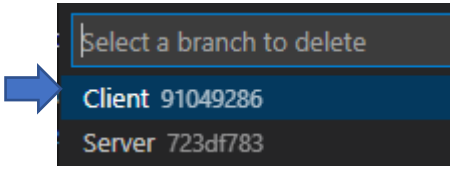


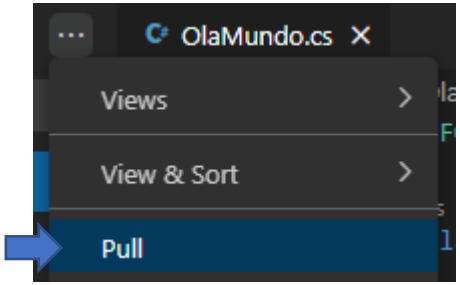
A branch “Client” não existe mais no GitHub (porque estou com head nela)

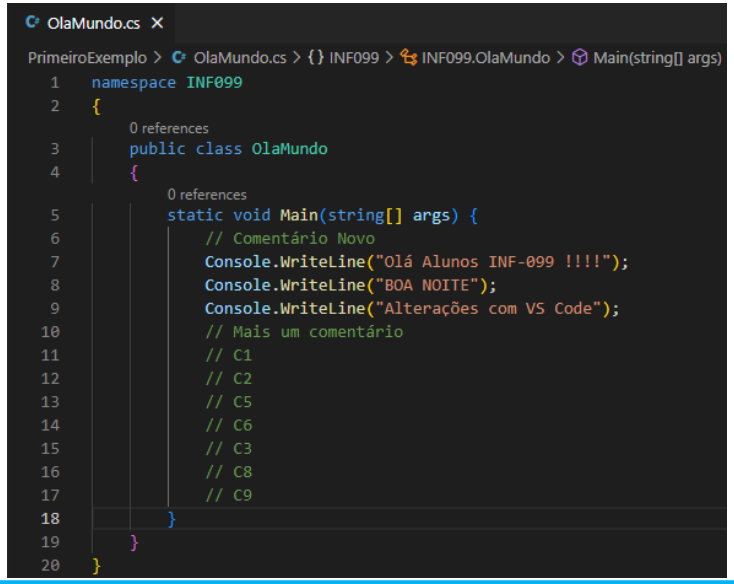
VS Code + GitHub

3) Fazer *rebase* da “client” na “master” (*main*)



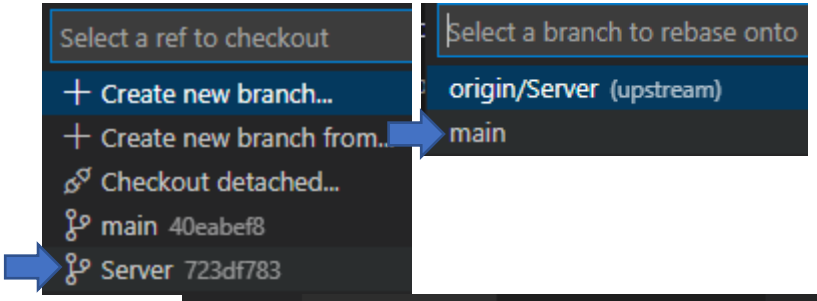




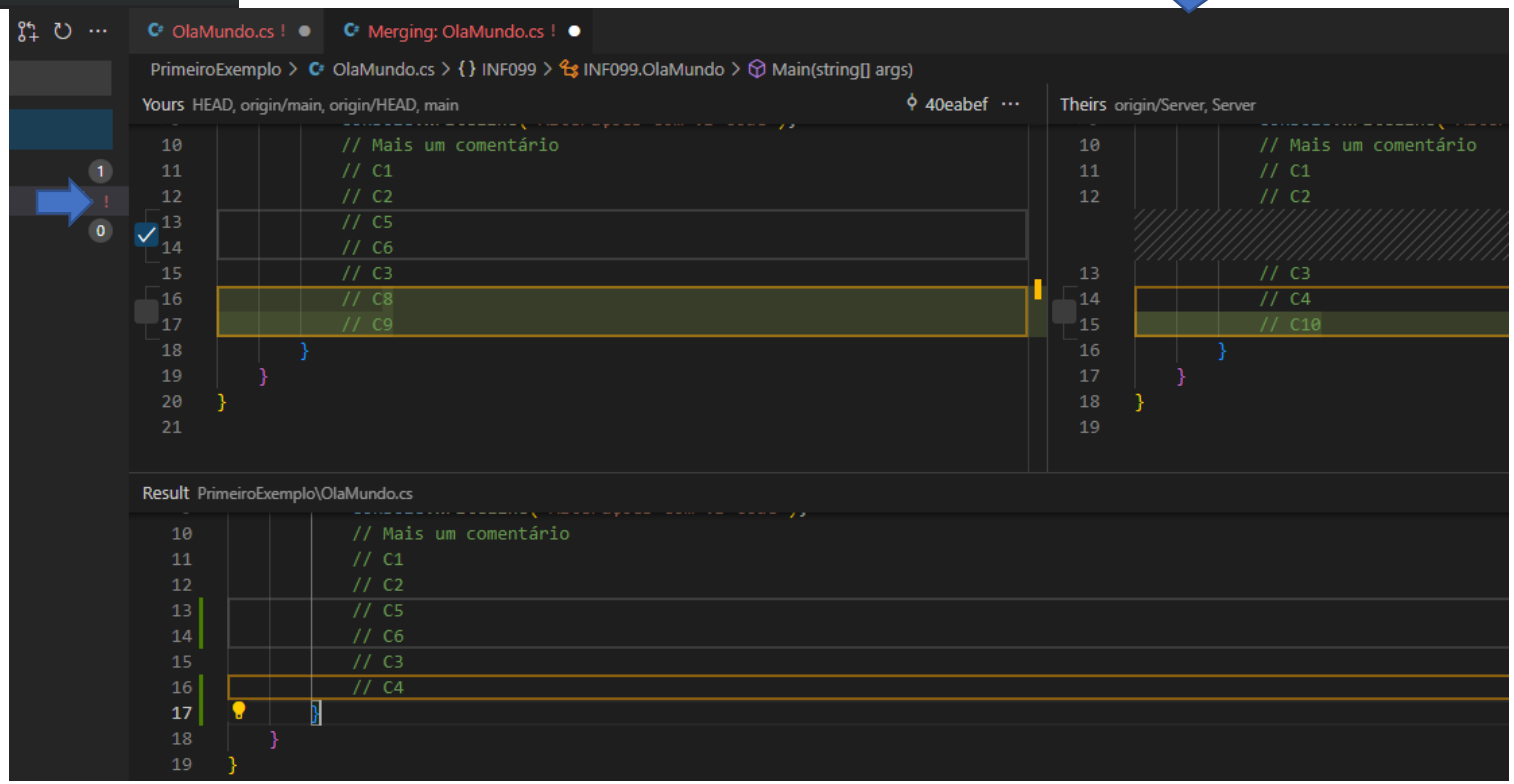


VS Code + GitHub

3) Fazer *rebase* da “server” na “master” (*main*) -

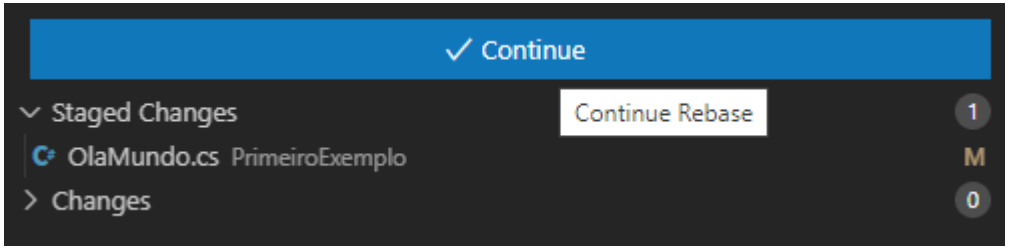


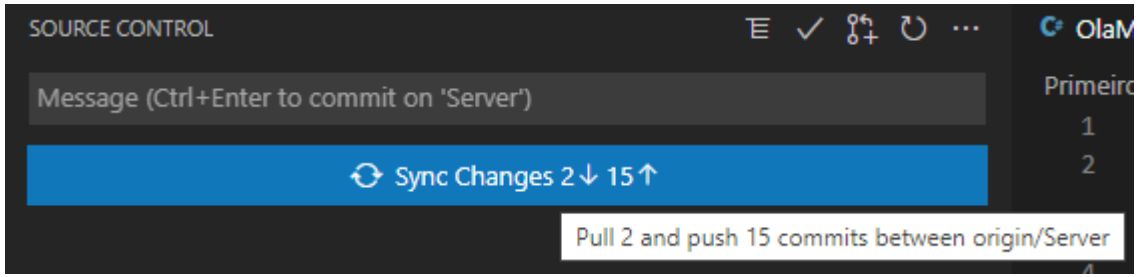
Ferramenta de Merge do VS Code

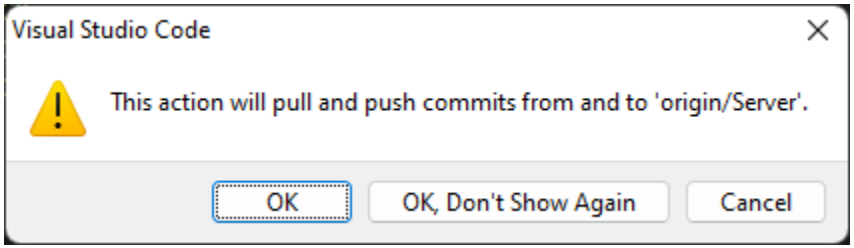


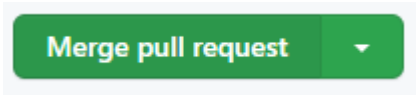
VS Code + GitHub

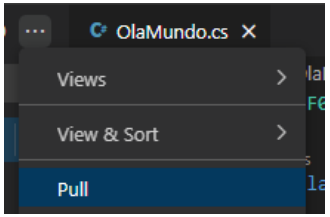
3) Fazer *rebase* da “server” na “master” (*main*) -

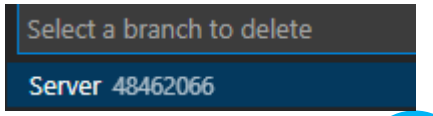






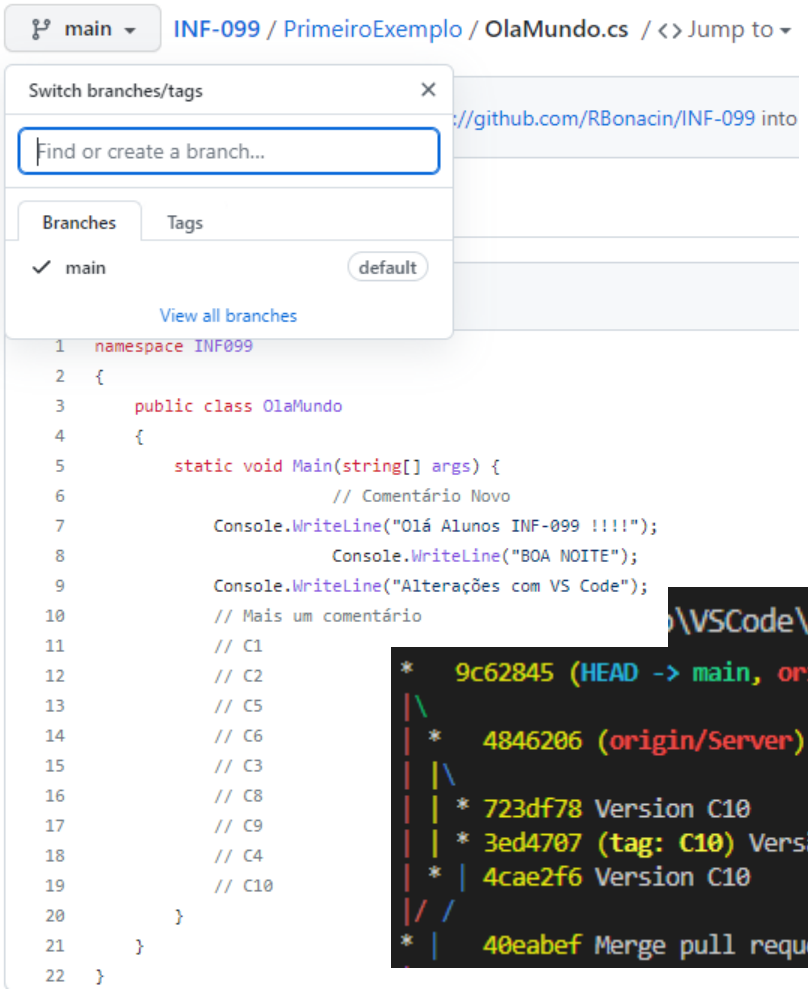






VS Code + GitHub

4) Verificar




```
\VSCode\INF-099> git log --oneline --decorate --graph --all
*   9c62845 (HEAD -> main, origin/main, origin/HEAD) Merge pull request #4 from RBonacin/Server
|\
| *   4846206 (origin/Server) Merge branch 'Server' of https://github.com/RBonacin/INF-099 into Server
| |\
| | * 723df78 Version C10
| | * 3ed4707 (tag: C10) Versão C4
| * | 4cae2f6 Version C10
|/ /
* |   40eabef Merge pull request #3 from RBonacin/Client
```

Trabalho 2

Trabalho 2

INF-0991 – Controle de Versão

Curso de Extensão Tecnologias Microsoft

Universidade Estadual de Campinas – UNICAMP

Prof. Rodrigo Bonacin

rbonacin@unicamp.br

03/setembro/2022

TRABALHO 2

1 Objetivos

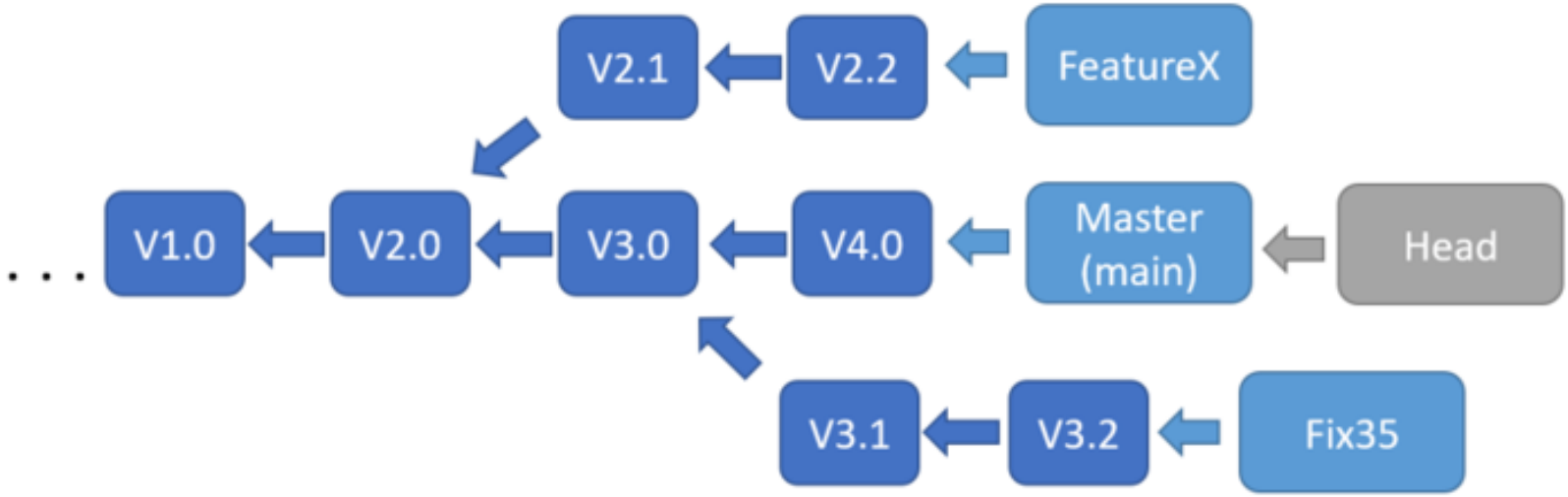
Este trabalho tem como objetivo desenvolver habilidades de uso de sistema de controle de versão "Git" via ferramentas do Visual Studio Code. Este trabalho compõe a avaliação da disciplina INF0991, e deverá ser realizado em grupos de duas até quatro pessoas.

Trabalho 2

2 Atividade

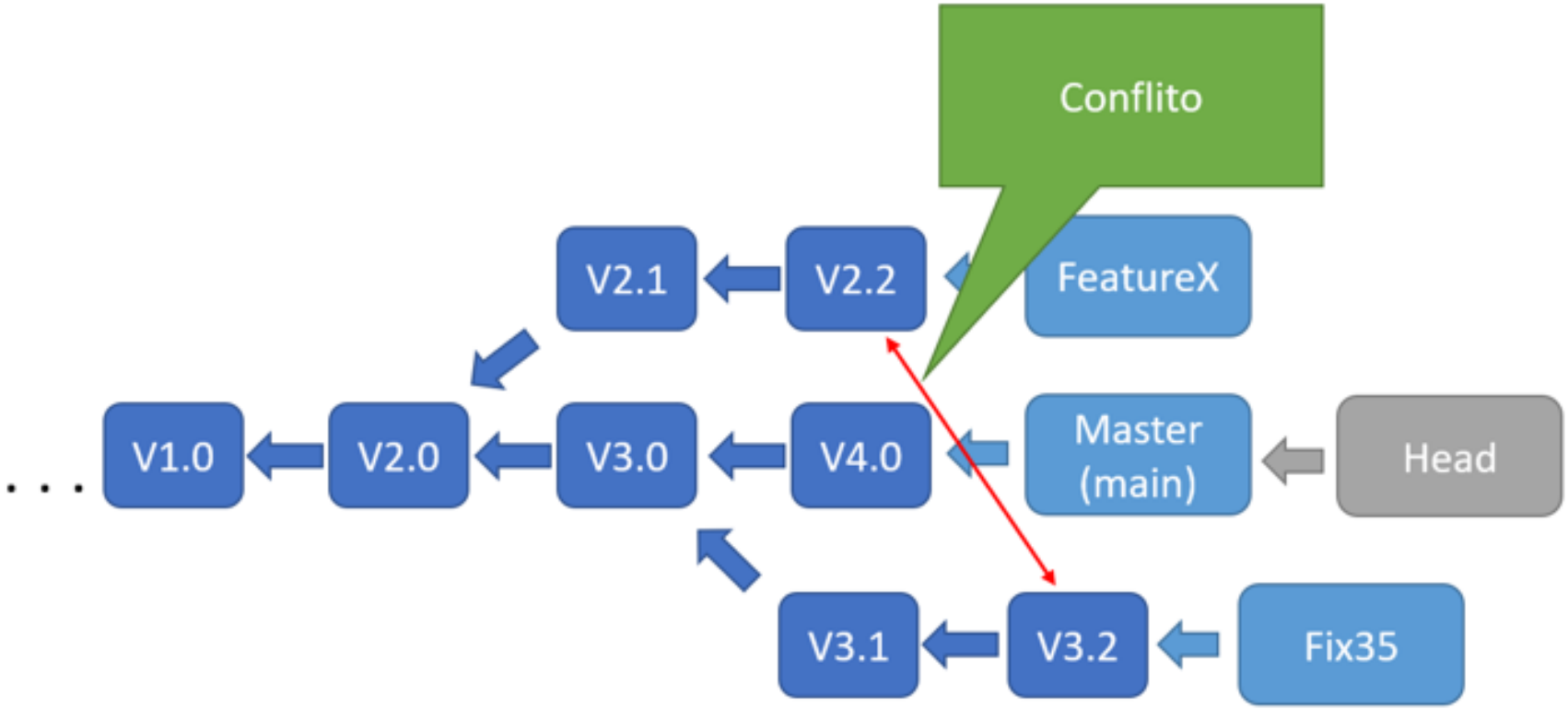
As seguintes tarefas devem ser realizadas na execução desta atividade.

1. Inserir código C# no GitHub de um membro do grupo (preferencialmente, código gerado na disciplina INF-0990 Programação em C#).
2. Inserir alterações via Visual Studio Code por diferentes membros da equipe (em *branches* separados) de modo que tenhamos a seguinte configuração (utilizar *tags* para identificar as versões):



Trabalho 2

3. Nesta configuração deve existir (pelo menos) um conflito entre as versões V2.2 e V3.2:



Trabalho 2

4. Fazer "*rebase*" dos *branches* "Fix35" e "FeatureX" para o "Master (main)" de modo a resultar na seguinte configuração:

5. Eliminar eventuais conflitos.
6. Atualizar o GitHub (*push*).
7. Verificar alterações no GitHub.

3 Entrega

Os seguintes itens devem ser entregues:

1. Link com repositório GitHub com históricos de alterações, desde a inclusão dos arquivos até a *rebase* e solução de conflitos.
2. Relatório com cada tela (*print*) da execução no repositório local com todos os comandos executados na interface do Visual Studio Code.
3. Print com log de alterações/*commits* no repositório local (*git log --oneline --decorate --graph --all*) a cada *commit*.
4. Ou seja, cada tarefa supracitada deve estar documentada de modo a ilustrar sua execução.

Deverá ser entregue um arquivo por grupo, constando os respectivos nomes e números de matrículas. As entregas deverão ser realizadas pelo Moodle.

DATA FINAL DE ENTREGA: 09/09/2022