

# Programação Orientada a Objetos

Victor Machado da Silva, MSc  
[victor.silva@professores.ibmec.edu.br](mailto:victor.silva@professores.ibmec.edu.br)

# Índice

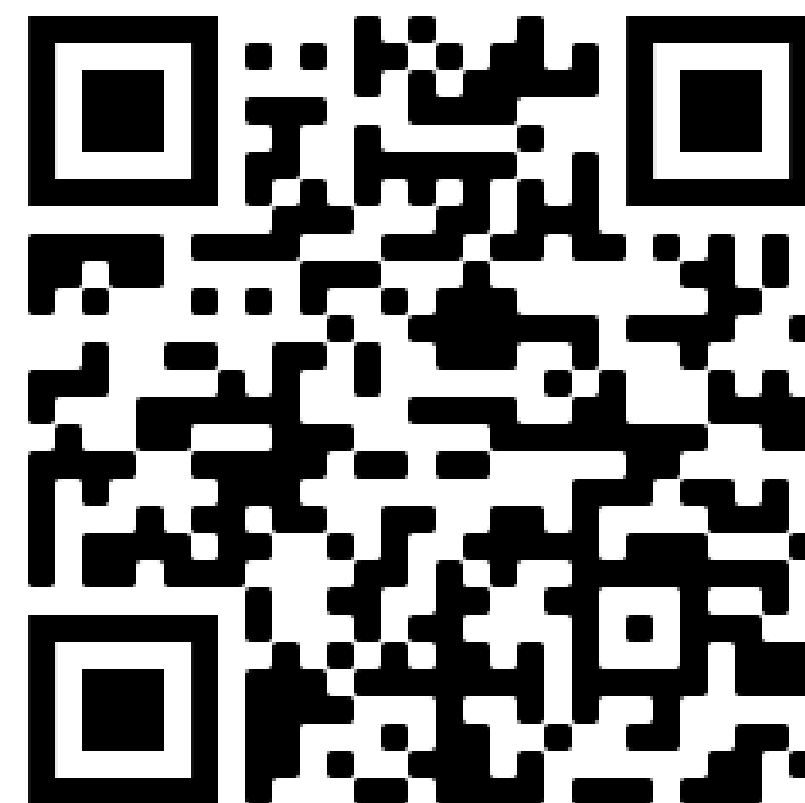
- [Apresentação do curso](#)
- [Configurando o ambiente](#)
- [Sobre paradigmas de programação](#)
- [Git](#)



# Apresentação do curso

# Apresentação do curso

- Contato: [victor.silva@professores.ibmec.edu.br](mailto:victor.silva@professores.ibmec.edu.br)
- Aulas às segundas e quartas-feiras, de 7:30 às 9:21
- Grupo no Whatsapp: <https://chat.whatsapp.com/BCQDk3VSc4f0BjN2vbtpsa>
- Material no GitHub: <https://github.com/victor0machado/2021.1-progoo>



# Apresentação do curso

Aula	Dia	Dia sem.	Tópico
01	22/02/2021	seg	Introdução à disciplina
02	24/02/2021	qua	Sobre paradigmas de programação / Introdução ao Git
03	01/03/2021	seg	Fundamentos de OO: abstração, reuso, encapsulamento
04	03/03/2021	qua	Conceitos iniciais de Java: parte 1
05	08/03/2021	seg	Conceitos iniciais de Java: parte 2
06	10/03/2021	qua	Conceitos iniciais de Java: parte 3
07	15/03/2021	seg	Conceitos estruturais: classe, atributo, método
08	17/03/2021	qua	Conceitos estruturais: objeto
09	22/03/2021	seg	Noções de UML: introdução
10	24/03/2021	qua	Noções de UML: diagramas de classes
11	29/03/2021	seg	Conceitos relacionais: herança e polimorfismo
12	31/03/2021	qua	Conceitos relacionais: associação
13	05/04/2021	seg	Conceitos relacionais: interface
14	07/04/2021	qua	SEM AULA (SEMANA AP1)
15	12/04/2021	seg	SEM AULA (SEMANA AP1)
16	14/04/2021	qua	SEM AULA (SEMANA AP1)
17	19/04/2021	seg	Conceitos organizacionais: pacotes
18	21/04/2021	qua	SEM AULA (TIRADENTES)
19	26/04/2021	seg	Conceitos organizacionais: visibilidades
20	28/04/2021	qua	Noções de UML: diagramas de caso de uso

# Apresentação do curso

Aula	Dia	Dia sem.	Tópico
21	03/05/2021	seg	Noções de UML: diagramas de atividades
22	05/05/2021	qua	Noções de UML: diagramas de sequência
23	10/05/2021	seg	Boas práticas da OOP: parte 1
24	12/05/2021	qua	Boas práticas da OOP: parte 2
25	17/05/2021	seg	Boas práticas da OOP: parte 3
26	19/05/2021	qua	Princípios SOLID: parte 1
27	24/05/2021	seg	Princípios SOLID: parte 2
28	26/05/2021	qua	Princípios SOLID: parte 3
29	31/05/2021	seg	Princípios SOLID: parte 4
30	02/06/2021	qua	Princípios SOLID: parte 5
31	07/06/2021	seg	Design smells
32	09/06/2021	qua	Métricas de código
33	14/06/2021	seg	Frameworks Java
34	16/06/2021	qua	Frameworks Java
35	21/06/2021	seg	Frameworks Java
36	23/06/2021	qua	SEM AULA (SEMANA AP2)
37	28/06/2021	seg	SEM AULA (SEMANA AP2)
38	30/06/2021	qua	SEM AULA (SEMANA AP2)
39	05/07/2021	seg	SEM AULA (SEMANA AS)
40	07/07/2021	qua	SEM AULA (SEMANA AS)

# Apresentação do curso

## Avaliação

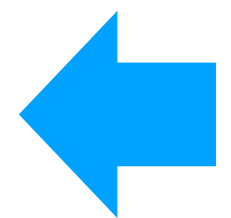
- Proporção:
  - Exercícios periódicos (AC): 20%
  - Projeto (AP1): 40%
  - Projeto (AP2): 40%
- Detalhes das entregas:
  - Exercícios da AC devem ser individuais
  - Projetos de AP1 e AP2 em grupos de no mínimo 2 e no máximo 3 pessoas
- AS será uma prova com consulta, que substituirá a menor nota entre AP1 e AP2.
- Os trabalhos serão *commitados* no GitHub em um *branch* separado do *master*, em um repositório privado criado pelos alunos, e um *pull request* deverá ser enviado para avaliação.



# Sugestões de materiais para estudo

- Apostila Java e Orientação a Objetos (Caelum - <https://www.caelum.com.br/apostilas>)
- Curso Java Completo (DevDojo - [https://www.youtube.com/watch?v=kkOSweUhGZM&list=PL62G310vn6nHrMr1tFLNOYP\\_c73m6nAzL](https://www.youtube.com/watch?v=kkOSweUhGZM&list=PL62G310vn6nHrMr1tFLNOYP_c73m6nAzL))
- Thiago Leite e Carvalho - Orientação a Objetos: Aprenda seus Conceitos e suas Aplicabilidades de Forma Efetiva (Casa do Código, 2016)
- Joshua Bloch - Java Efetivo: As Melhores Práticas para a Plataforma Java (Alta Books, 2019)
- Maurício Aniche - Orientação a Objetos e SOLID para Ninjas: Projetando Classes Flexíveis (Casa do Código, 2015)



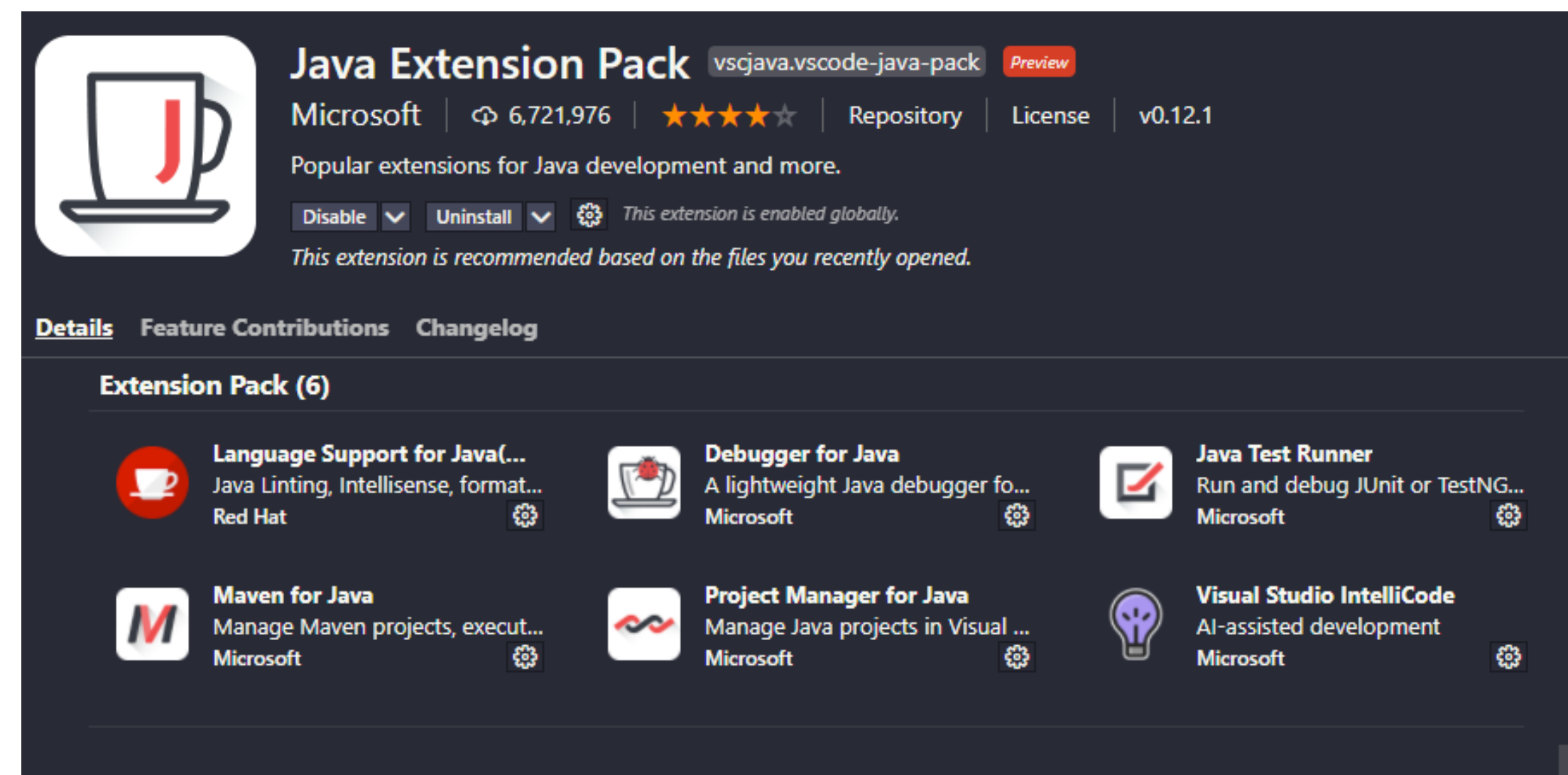


# Configurando o ambiente

# Configurando o ambiente

## IDE:

- Neste curso usaremos o **VSCode** como IDE principal para o desenvolvimento de código. Caso você ainda não tenha, sugiro dar uma olhada nos slides da disciplina de Programação (link [aqui](#)) e fazer a instalação e configuração inicial do software.
- Dentro do VSCode, será necessário instalar a *Java Extension Pack*, um pacote organizado pela própria Microsoft, com diversas extensões específicas para Java.



# Configurando o ambiente

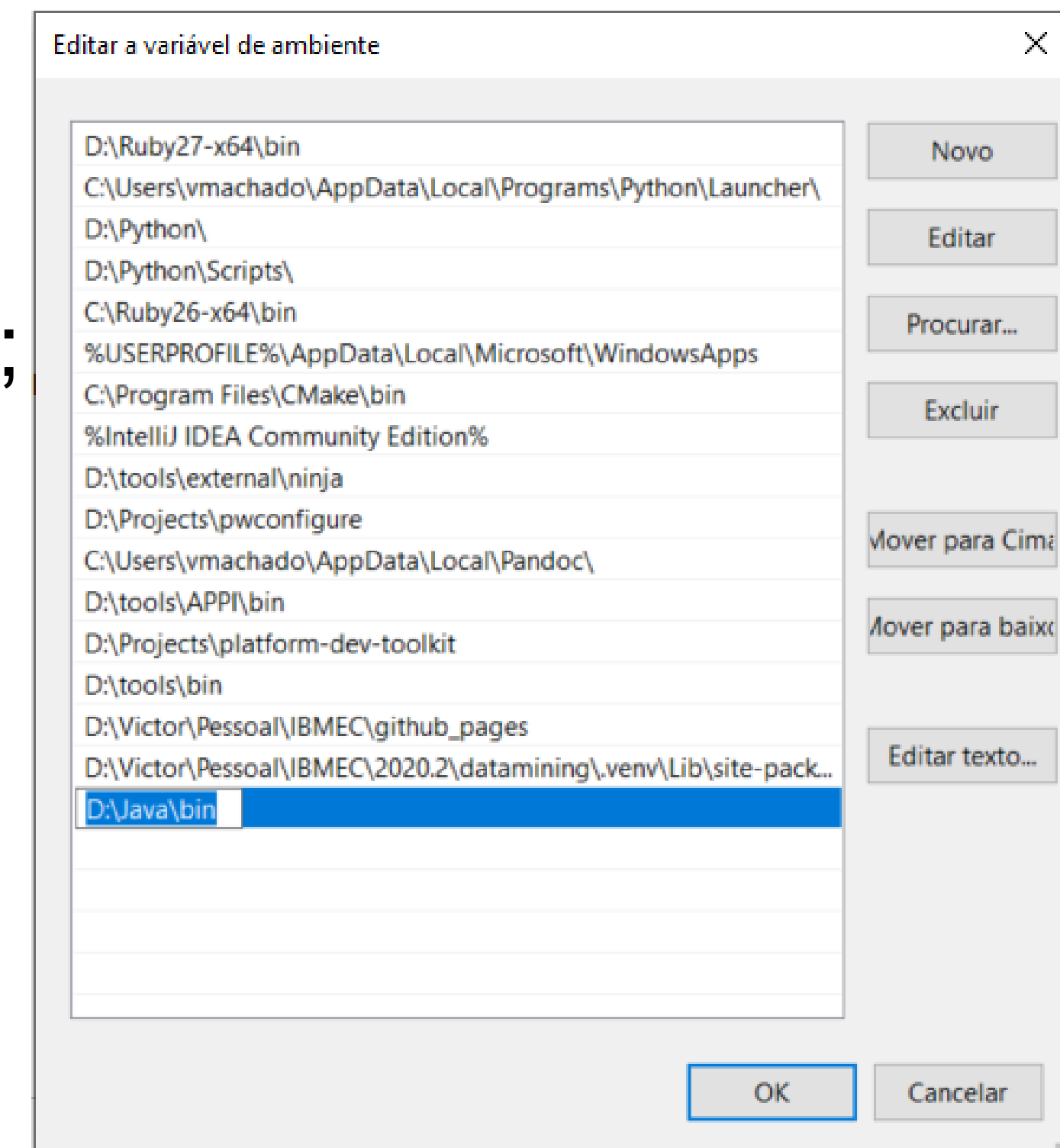
## Java:

- Normalmente, todo mundo possui o Java instalado no computador. No entanto, caso você não tenha, clique [aqui](#) para acessar o site do Java, baixar o JRE (*Java Runtime Environment*) e instalá-lo;
- Além do JRE, será necessário instalar o JDK (*Java Development Kit*), plataforma de desenvolvimento para Java. Acesse [esse link](#), baixe a versão mais recente do Java SE (atualmente, é a versão 15) e instale o software;
- Como sugestão, recomendo instalar em um caminho curto, como por exemplo *D:\Java*, para facilitar o acesso. Após a instalação, coloque a pasta “bin” do diretório Java no seu PATH, para poder acessar o compilador pela linha de comando (ver mais informações no próximo slide).

# Configurando o ambiente

Como adicionar um caminho ao PATH:

- Aperte a tecla do Windows e pesquise por “Editar as variáveis de ambiente para sua conta”;
- Na janela que abrir, procure pela linha com “Path”, selecione-a e clique em “Editar...”;
- Clique em “Novo” e insira o caminho desejado;
- Aperte “Ok” e “Ok” novamente para fechar tudo;
- Para confirmar se está tudo certo, abra um prompt de comando novo e chame um arquivo ou executável que está contido na pasta (no caso da pasta “bin”, chame “javac”)



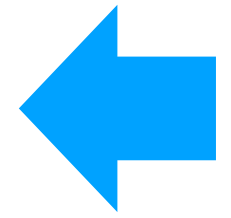
# Configurando o ambiente

## Git:

- Para o desenvolvimento dos trabalhos e acompanhamento da disciplina, vamos utilizar o Git para versionar os materiais;
- Para instalar o Git na máquina, acesse [esse site](#) e baixe o instalador. O processo de instalação é direto, basta clicar em “Next” para concluir a instalação;
- Também será necessário criar uma conta no GitHub ([www.github.com](https://www.github.com)). Caso queira, já pode me procurar por lá (victor0machado).

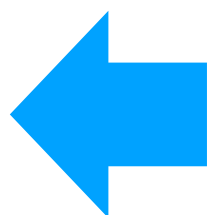
## Outros programas:

- Recomendo instalar o [notepad++](#) para um bom editor de texto simples.



# Sobre paradigmas de programação





# Git

# Introdução

- Talvez você reconheça esse nome por causa de sites como *github* ou *gitlab*. **Git** é um sistema *open-source* de controle de versionamento.
- Versionar projetos é uma prática essencial no mundo profissional e, em particular, na área de tecnologia, para manter um histórico de modificações deles, ou poder reverter alguma modificação que possa ter comprometido o projeto inteiro, dentre outras funcionalidades que serão aprendidas na prática.
- Os projetos são normalmente armazenados em **repositórios**.
- Vamos aqui falar de alguns conceitos principais sobre como funciona o sistema, independente da plataforma que vamos utilizar (p.ex., *github*). Em seguida vamos aplicar alguns desses conceitos na prática.

# Para mais informações...

- O tempo que temos em aula não é suficiente para conseguirmos discutir todos os detalhes por trás dessa tecnologia. Portanto, seguem abaixo algumas sugestões de conteúdos extras para estudarem:
  - <http://git-scm.com/book/en/Getting-Started-About-Version-Control>
  - <http://git-scm.com/book/en/Getting-Started-Git-Basics>
  - <http://learngitbranching.js.org/>
  - <http://try.github.io/levels/1/challenges/1>

# O que é Controle de Versões?

- Controle de Versões é um sistema de grava mudanças aplicadas em um arquivo ou um conjunto de arquivos ao longo do tempo, para que o usuário possa lembrar ou recuperar depois.
- Na área de tecnologia o controle de versões já é algo consolidado, uma vez que inúmeras alterações são aplicadas em um software durante a sua implementação e manutenção. No entanto, adotar um sistema de controle de versões (VCS, da sigla em inglês) é algo recomendado para qualquer área.
- Quando se quer adotar um VCS, normalmente o primeiro passo é trabalhar com um controle local, fazendo cópias dos arquivos e os renomeando com algum padrão (p.ex., incluindo a data ao final do nome do arquivo).
- O Git veio para otimizar esse controle de versões, permitindo inúmeras alterações que seriam muito complicadas se fossem feitas manualmente.

# Uma breve história do Git

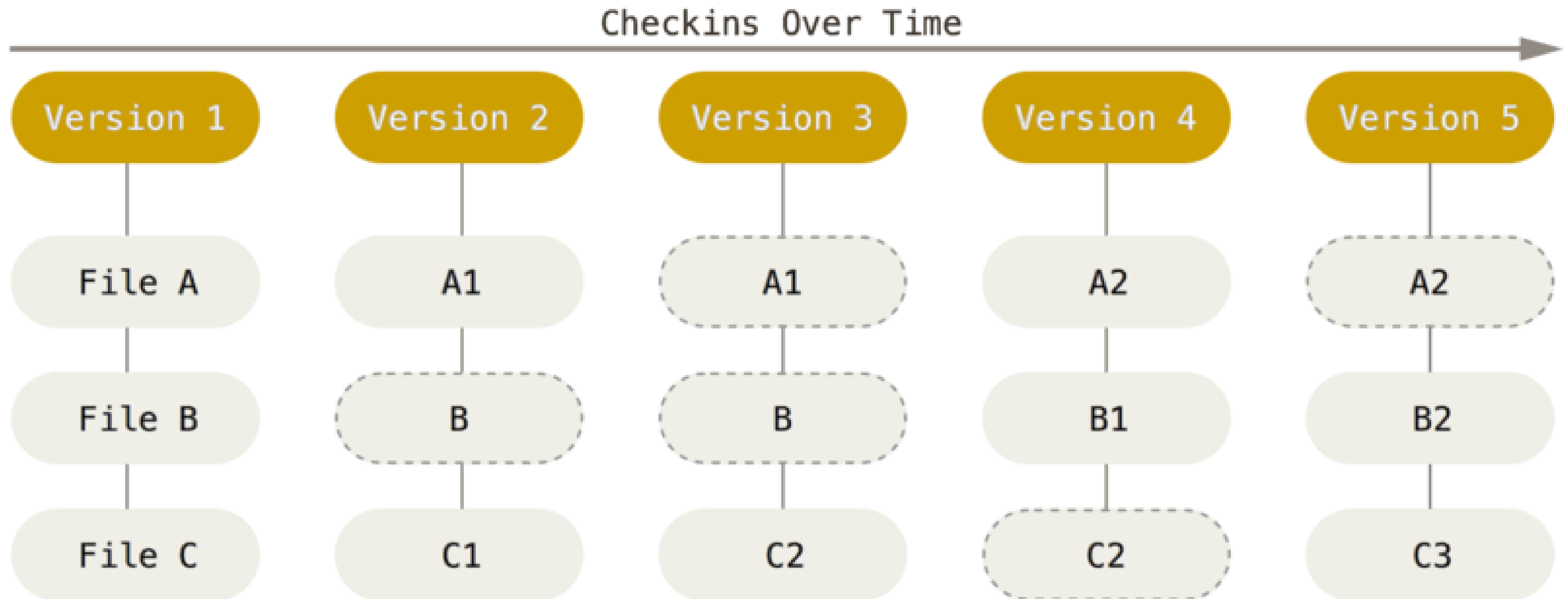
- A criação do Git está atrelada à criação do Linux, quando o time responsável utilizou uma solução de controle de versão que, eventualmente, tornou-se paga.
- A comunidade Linux, portanto, começou a planejar um sistema que aproveitasse algumas das características da solução anterior, porém evoluindo diversos aspectos.
- A comunidade tinha os seguintes objetivos para o novo sistema:
  - Velocidade
  - Design simples
  - Suporte forte para desenvolvimento não-linear, com milhares de atividades sendo realizadas em paralelo
  - Completamente distribuído, permitindo o acesso de qualquer lugar
  - Eficiente no suporte a grandes projetos

# O que é Git?

- **Git** funciona pensando que os dados de um repositório compõem uma série de “fotografias” de um sistema de arquivos em miniatura.
- No Git, a cada vez que você aplica uma alteração (ou *commit*), ou salva o estado do seu projeto, o Git basicamente tira uma “foto” de como os arquivos do repositório estão naquele momento e então ele armazena uma referência a essa foto.
- Para ser eficiente, se os arquivos não foram alterados, o Git não altera os arquivos novamente, apenas um link para a última versão do arquivo armazenada.
- Sendo assim, o Git trabalha os dados como um **fluxo de fotografias**.



# O que é Git?



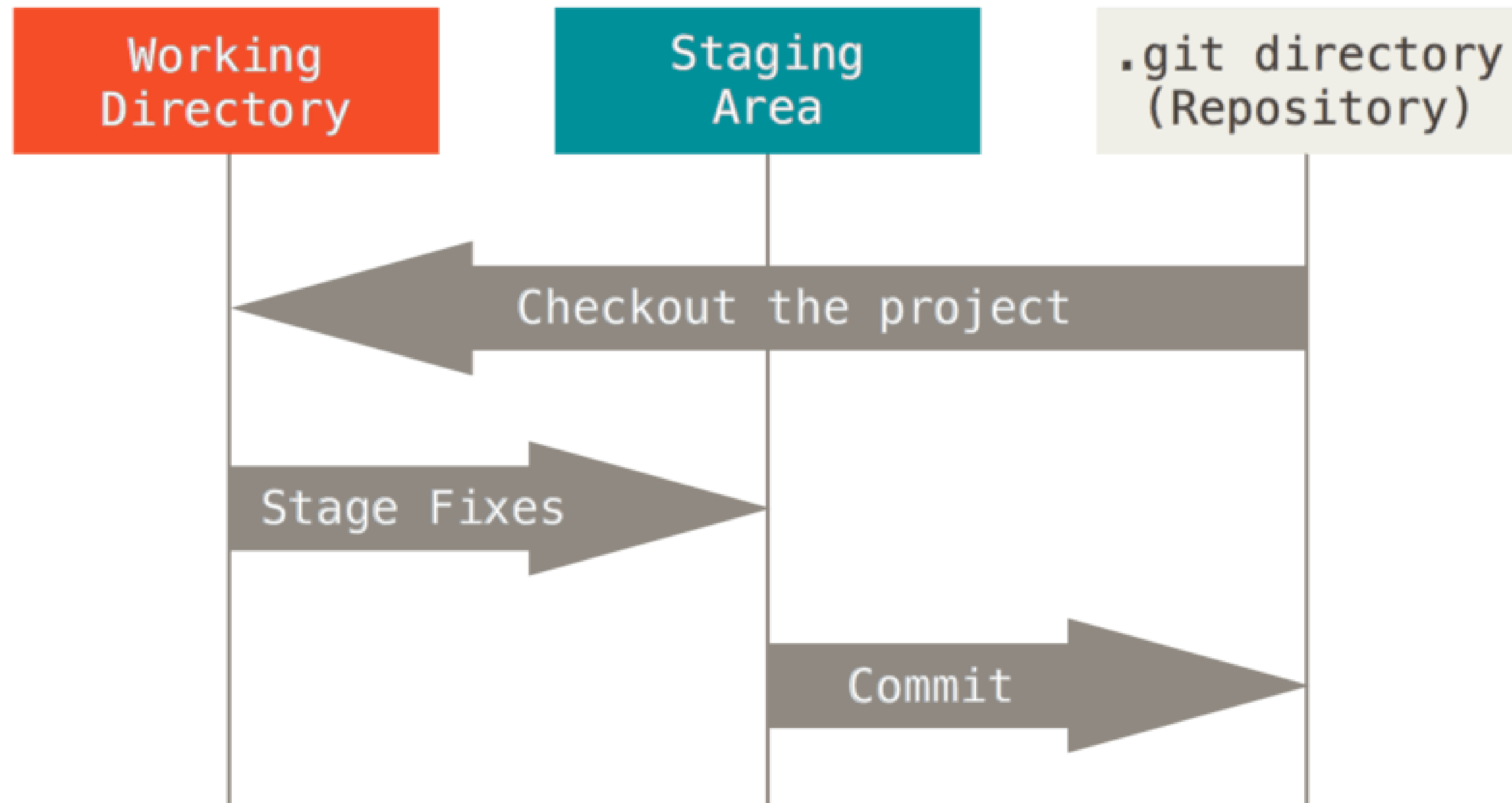
# O que é Git?

- A maior parte das operações no Git precisa apenas de arquivos e recursos locais para operarem, e normalmente nenhuma informação é necessária de outro computador na rede. Isso fornece uma velocidade de operação que outros VCS não possuem. Como cada usuário possui todo o histórico do projeto no computador, a maioria das operações aparenta ser quase instantânea.
- Para todos os efeitos, na prática, o Git normalmente só acrescenta informações ao seu banco de dados, nunca removendo informações. É muito difícil, e não recomendado, gerar operações que removam informações, já que essas operações podem afetar o histórico do seu projeto.

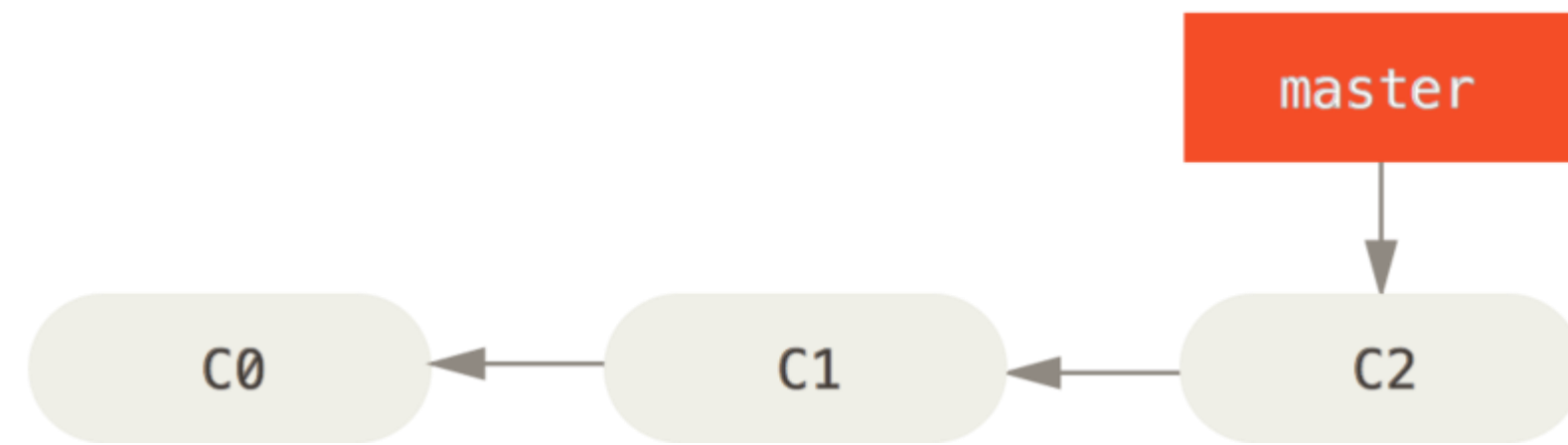
# Os três estados

- O Git tem três estados principais nos quais os arquivos de um repositório podem se encontrar: **modificado**, **preparado** e “**commitado**”:
  - **Commitado** significa que os dados estão armazenados de forma segura em seu banco de dados local;
  - **Modificado** significa que você alterou o arquivo, mas ainda não fez o commit no banco de dados;
  - **Preparado** significa que você marcou a versão atual de um arquivo modificado para fazer parte do seu próximo commit.
- Isso leva a três seções principais de um projeto Git: o diretório Git, o diretório de trabalho e área de preparo.

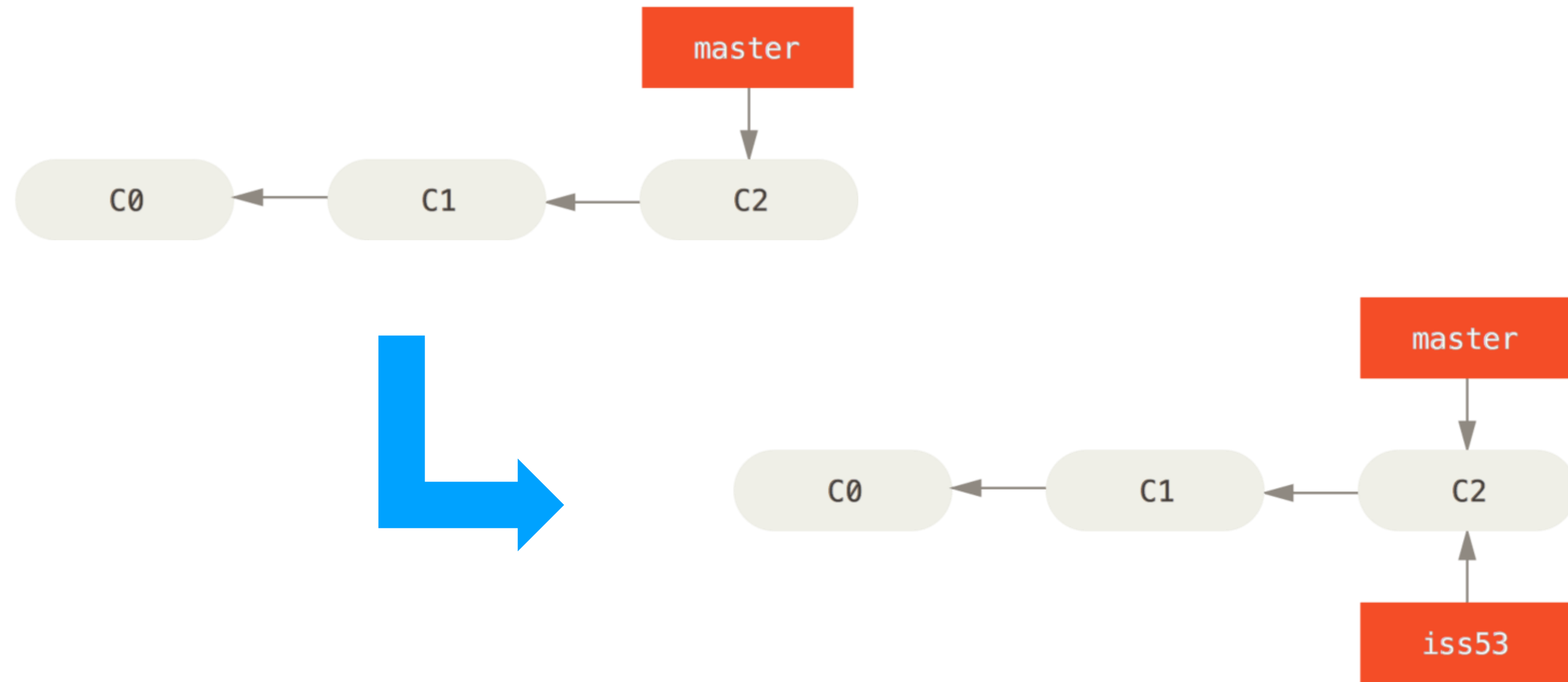
# Os três estados



# Branches no Git

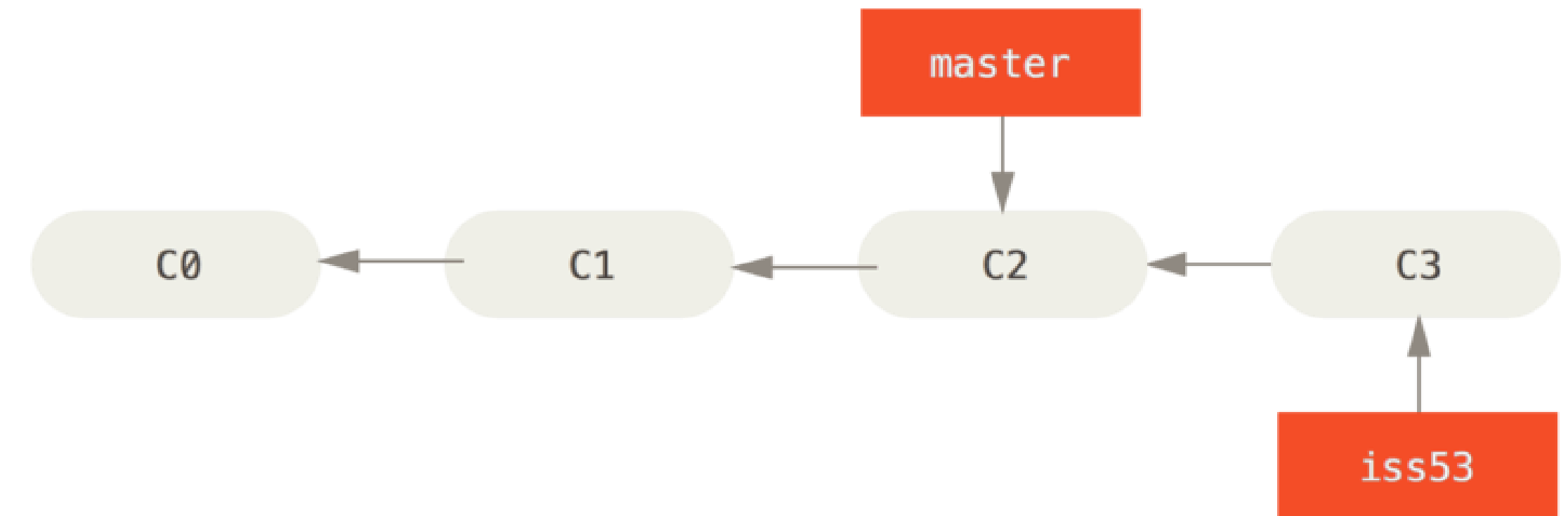
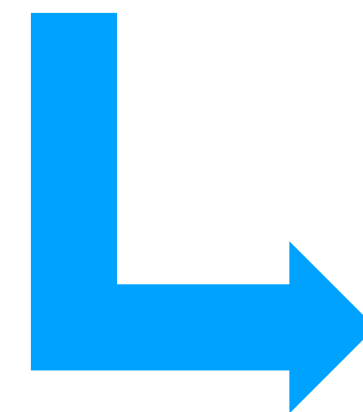
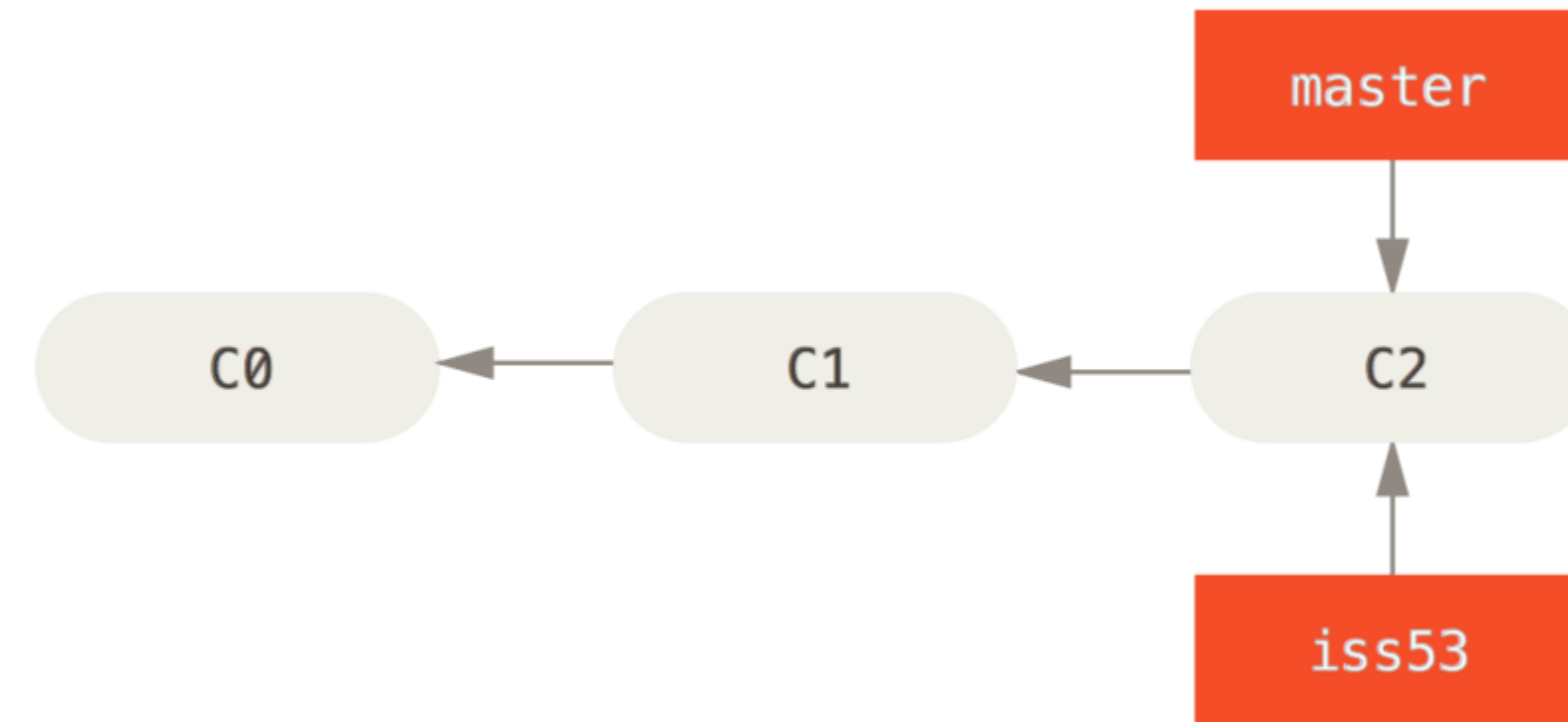
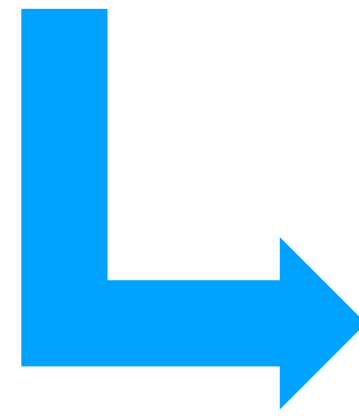
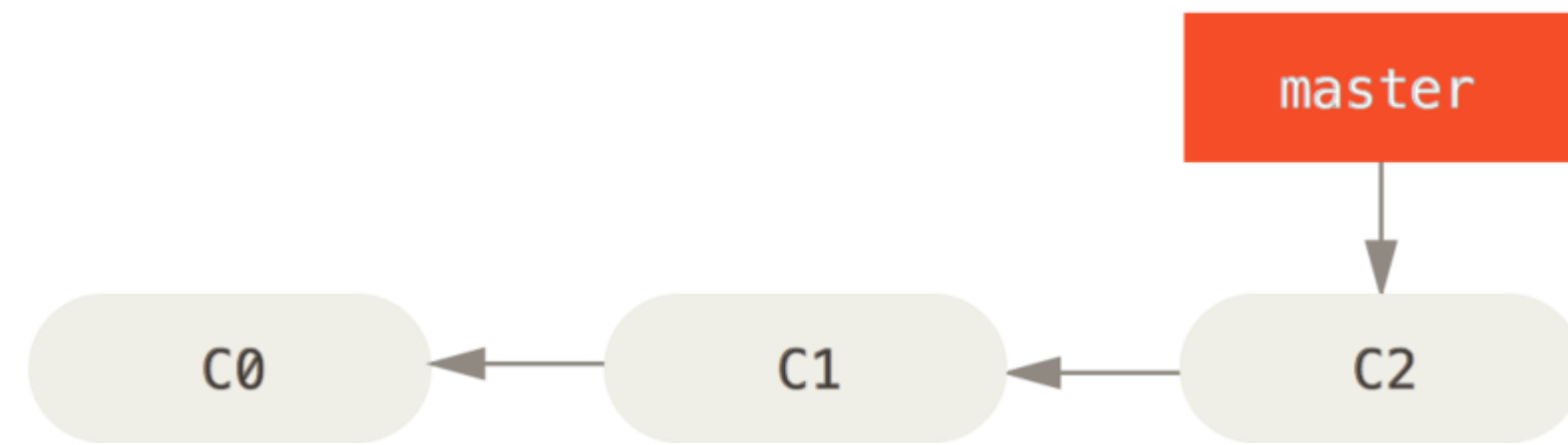


# Branches no Git

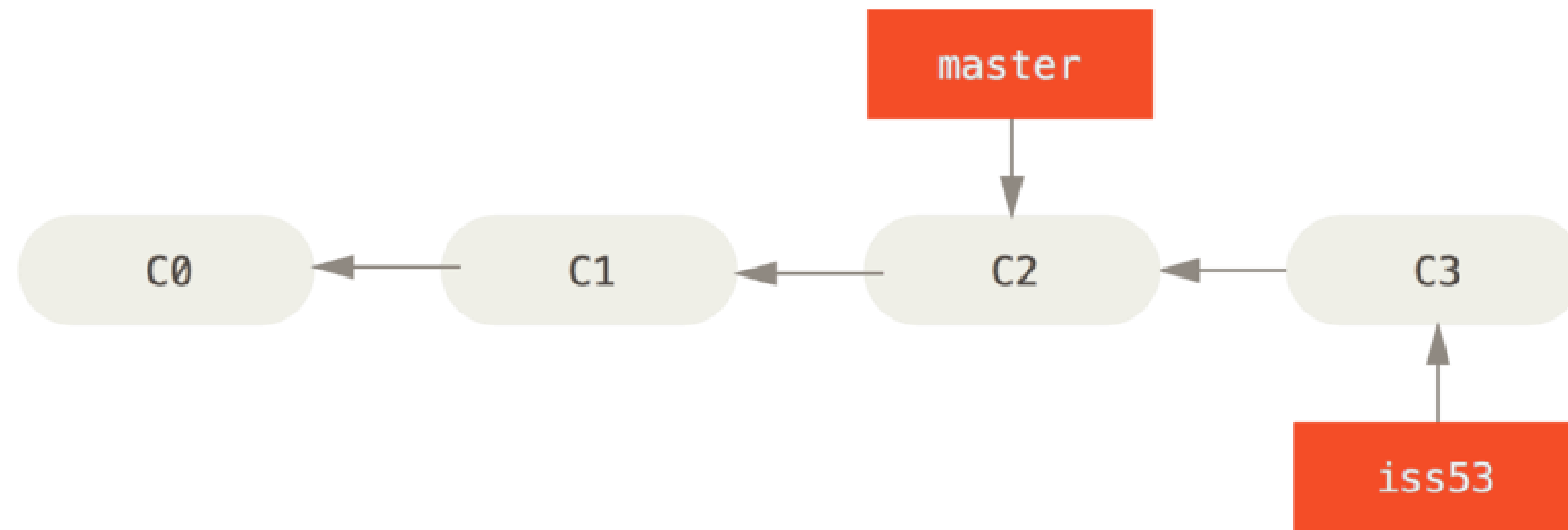




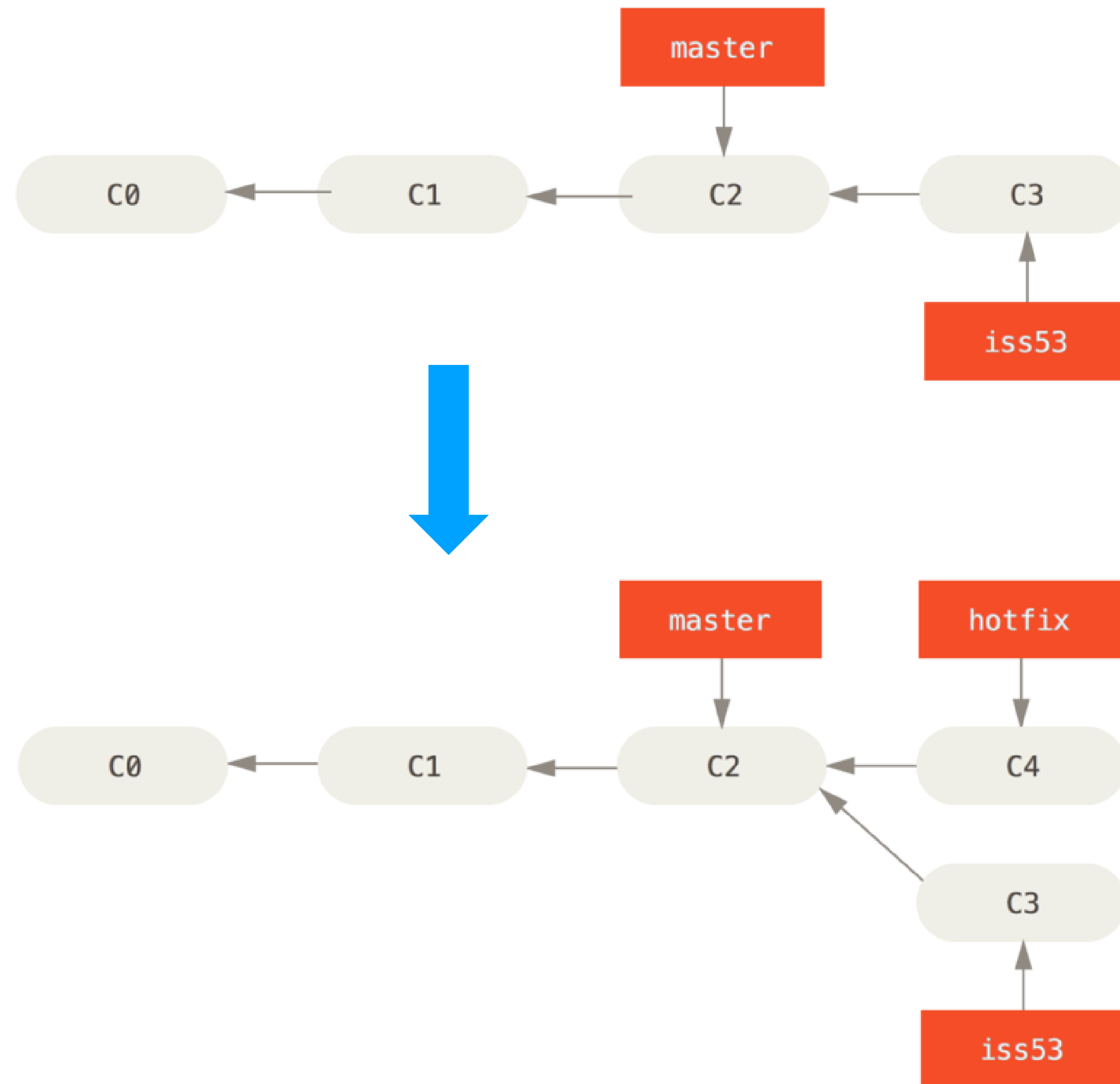
# Branches no Git



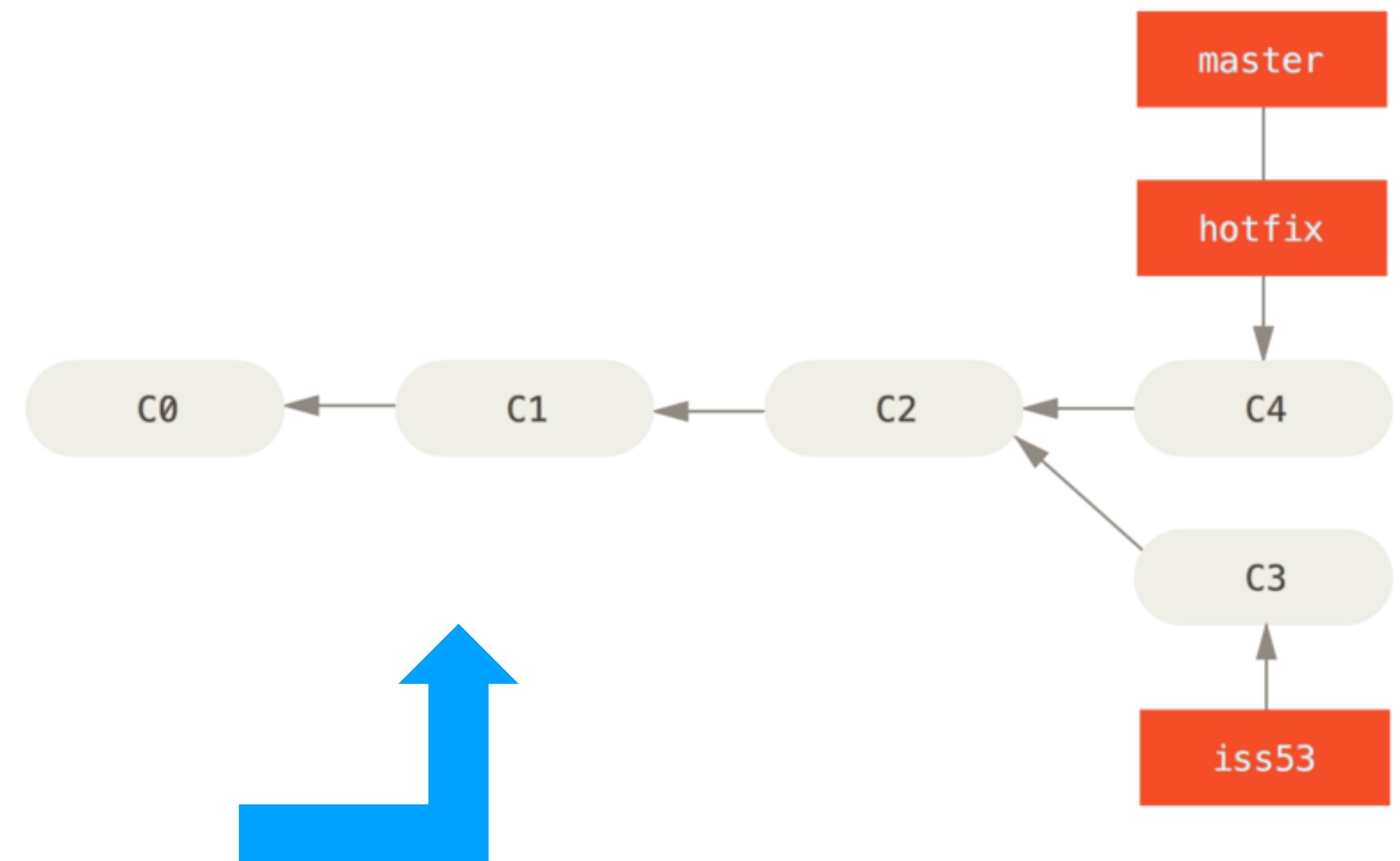
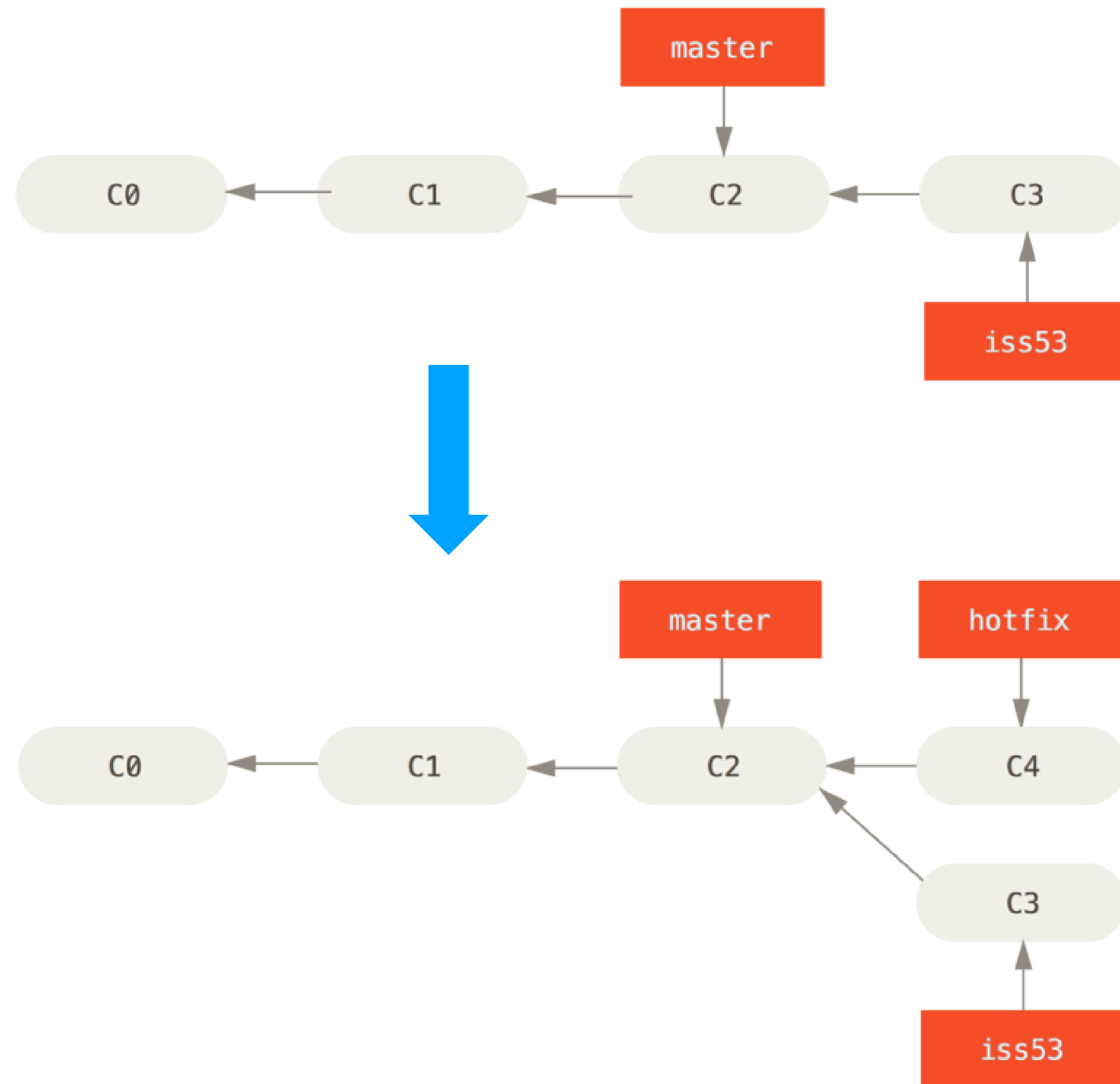
# Branches no Git



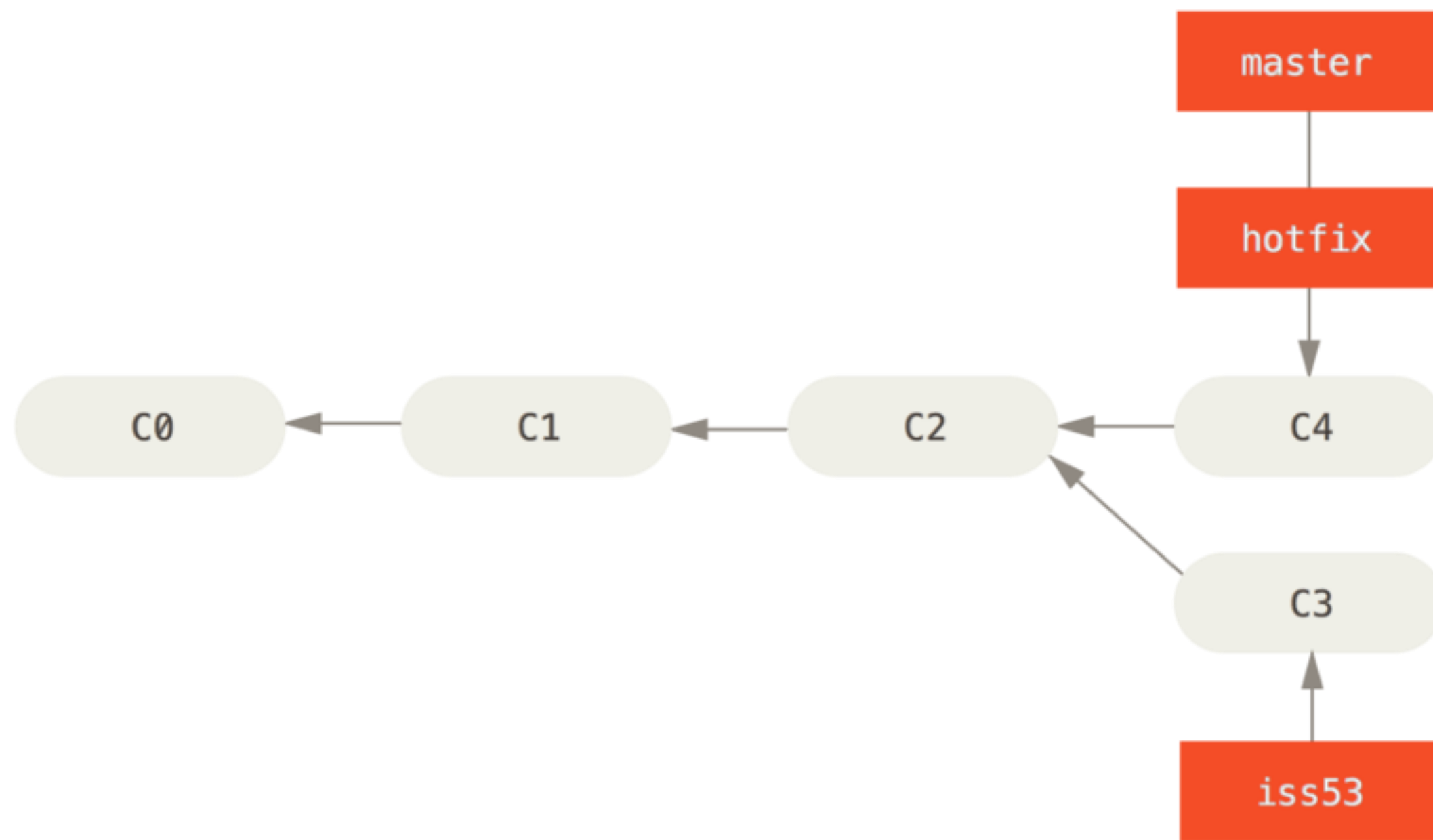
# Branches no Git



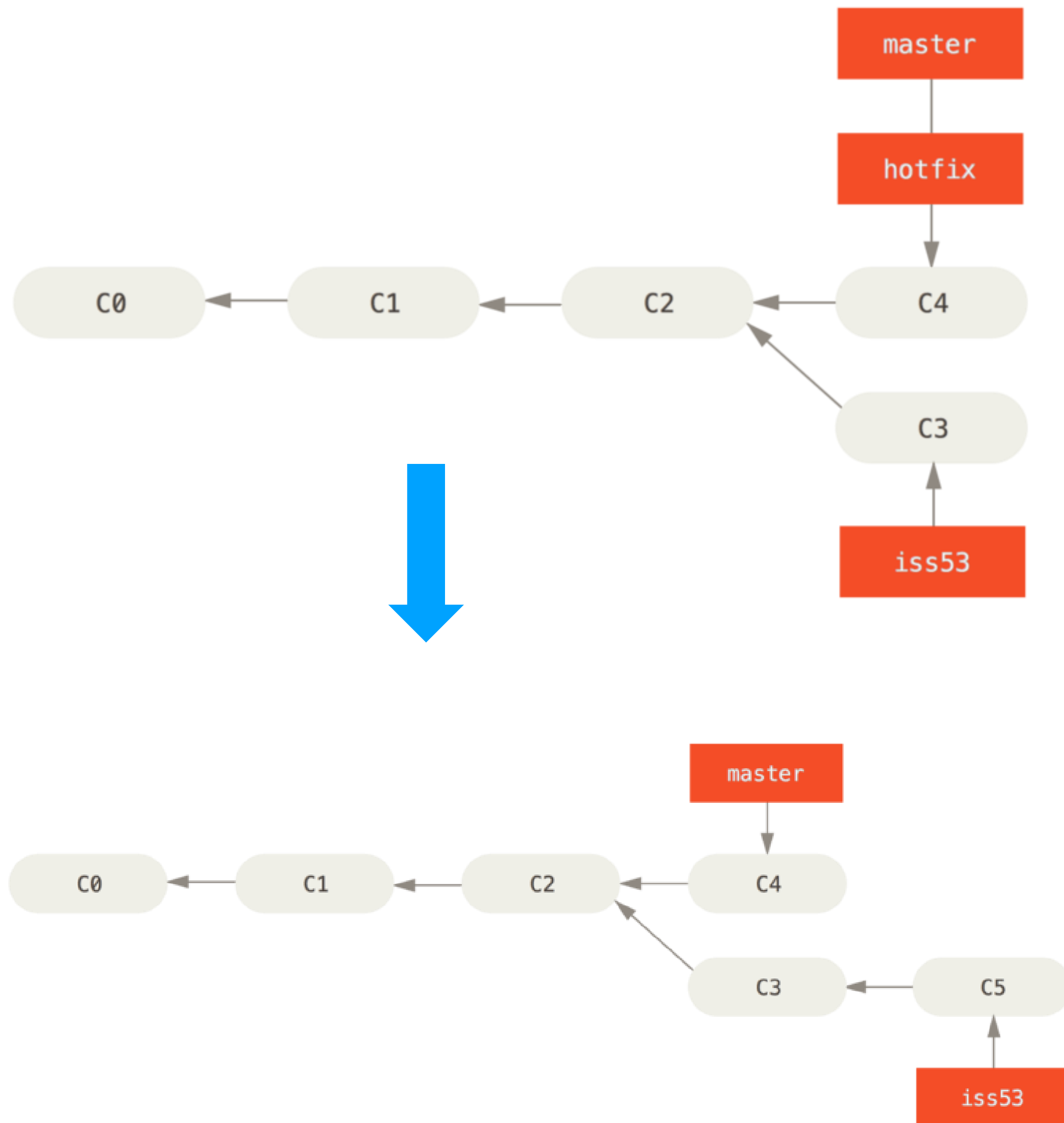
# Branches no Git



# Branches no Git

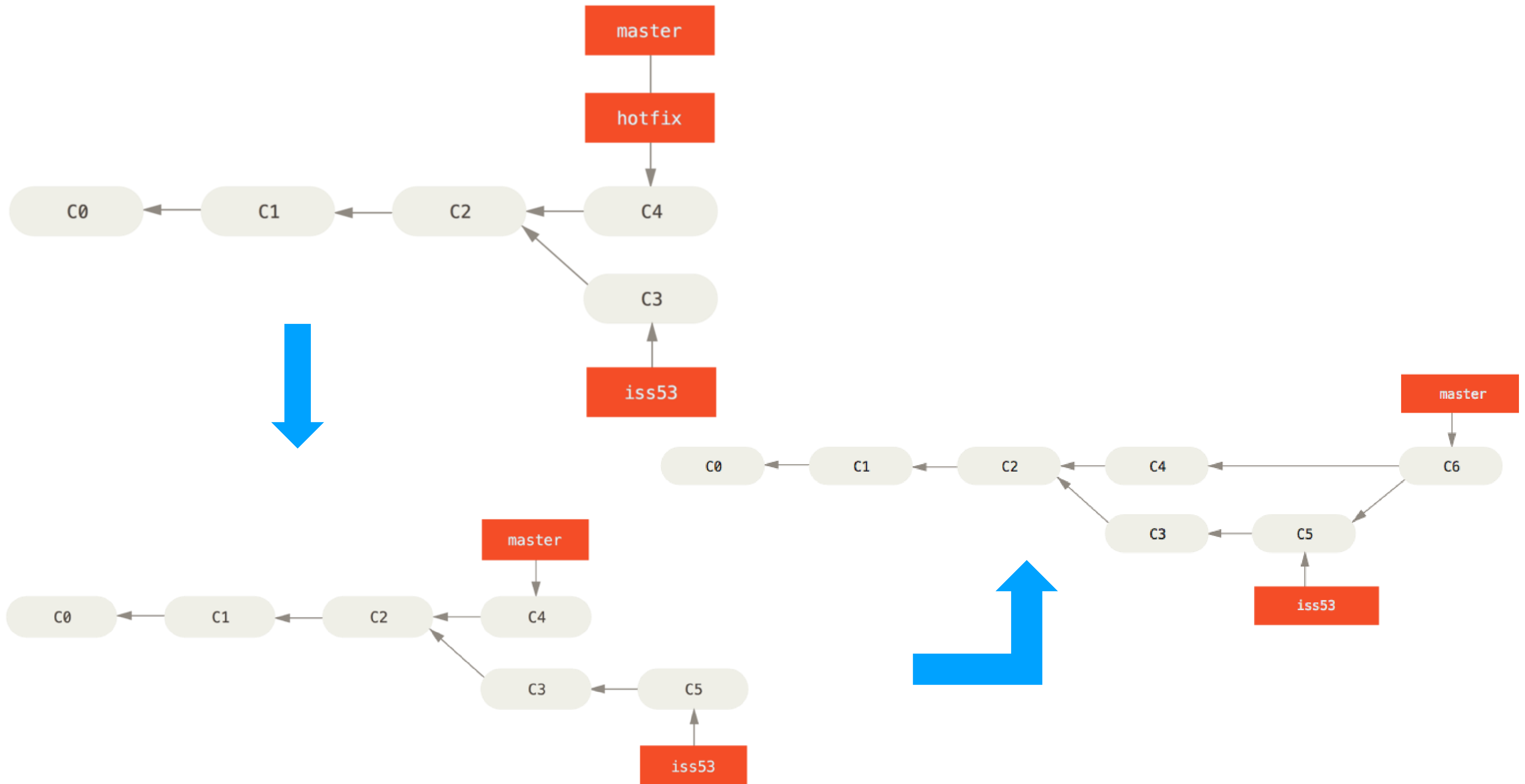


# Branches no Git





# Branches no Git



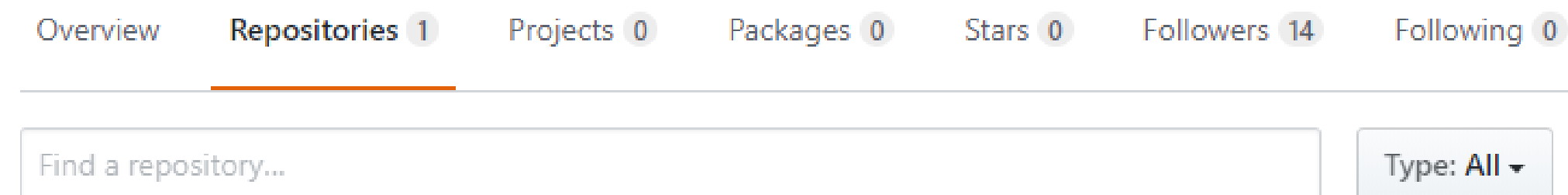
# Iniciando o uso de Git com GitHub

- Antes de mais nada, precisamos instalar o sistema Git na nossa máquina.
- <https://git-scm.com/downloads>
- O GitHub ([www.github.com](http://www.github.com)) é uma plataforma de versionamento que utiliza Git como base.
- Não é o nosso objetivo ensinar a plataforma a fundo, mas vamos utilizar algumas funcionalidades. Caso tenha interesse em se aprofundar no assunto, recomendo algumas páginas:
  - <https://github.com/culturagovbr/primeiros-passos>;
  - <https://help.github.com/en>;
  - [https://rogerdudler.github.io/git-guide/index.pt\\_BR.html](https://rogerdudler.github.io/git-guide/index.pt_BR.html).
- Antes de começar a configurar, crie uma conta no site.

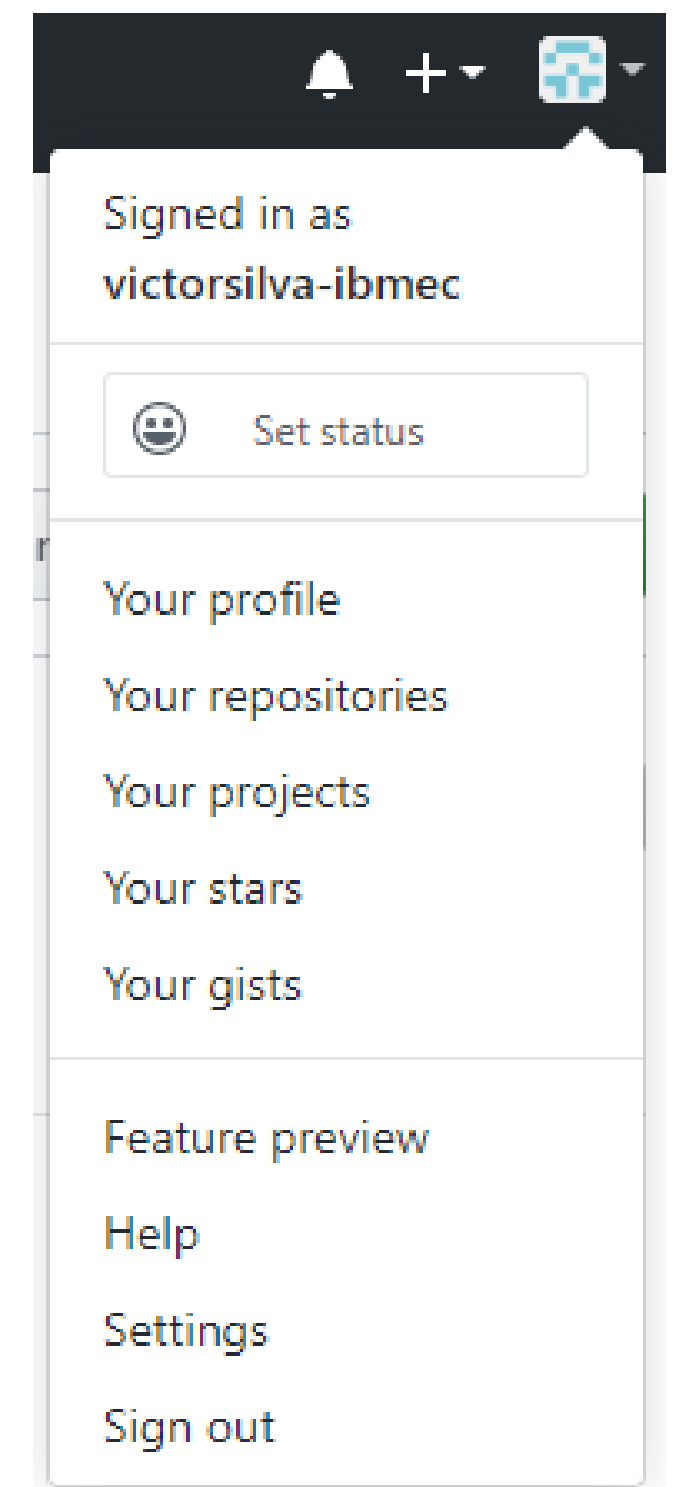
# Algumas tarefas no GitHub

- **Criando um novo repositório:**

- No canto superior direito, clique no ícone do seu usuário e, em seguida, em **Your Profile**;
- Na nova janela, clique em **Repositories** e em **New**;

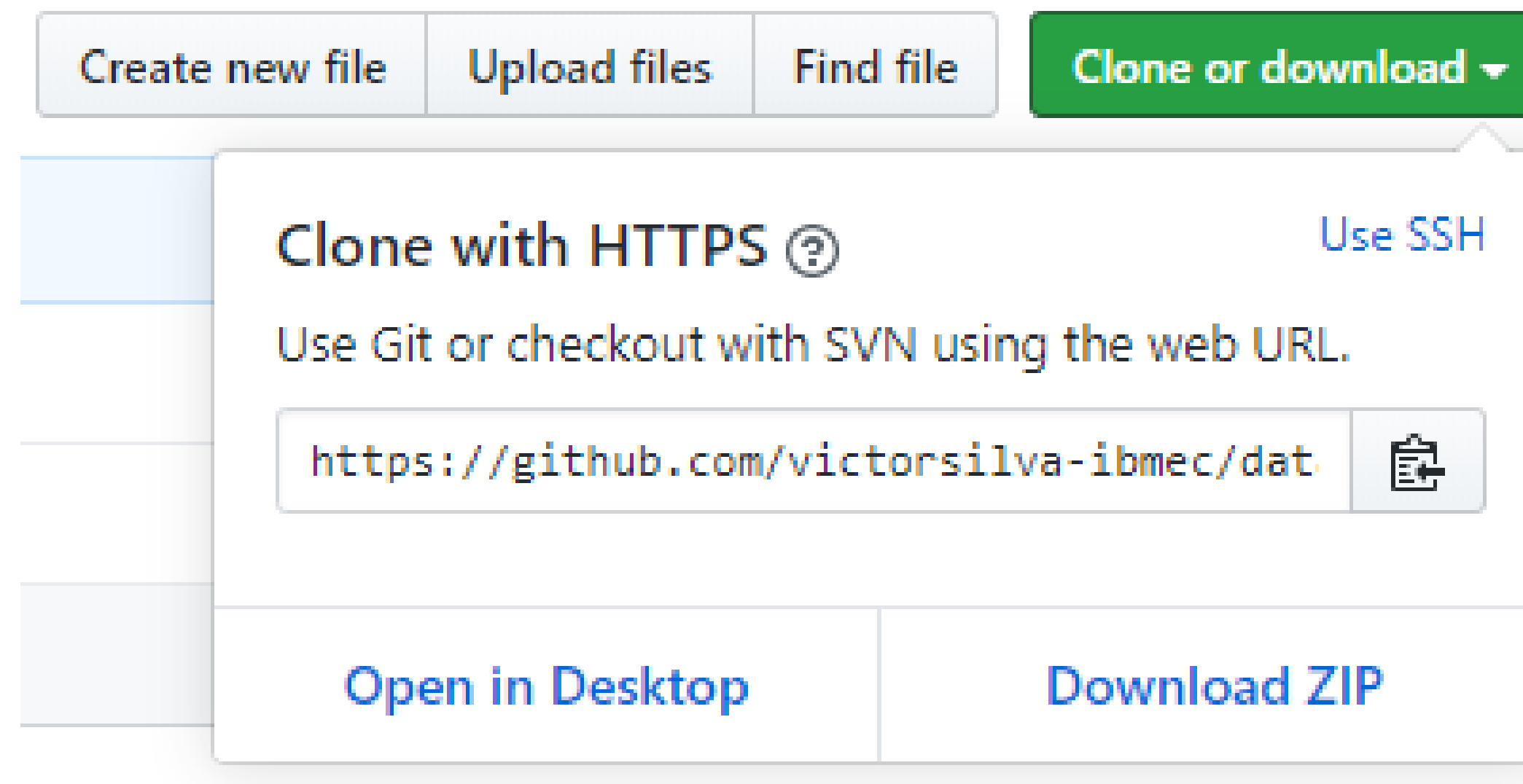


- Defina um nome (evite espaços e números), insira uma descrição se desejar e marque a opção **Public**, para que ele possa ser compartilhado. Por fim, marque a caixa **Initialize this repository with a README**;
- Clique em **Create repository**.

A screenshot of the GitHub 'Create new repository' form. The 'Owner' is 'victorsilva-ibmec' and the 'Repository name' is 'data-mining'. The 'Description' is 'Repositório para curso de Data Mining com Python'. The 'Public' option is selected. The 'Initialize this repository with a README' checkbox is checked. The 'Add .gitignore' and 'Add a license' buttons are at the bottom, along with a green 'Create repository' button.

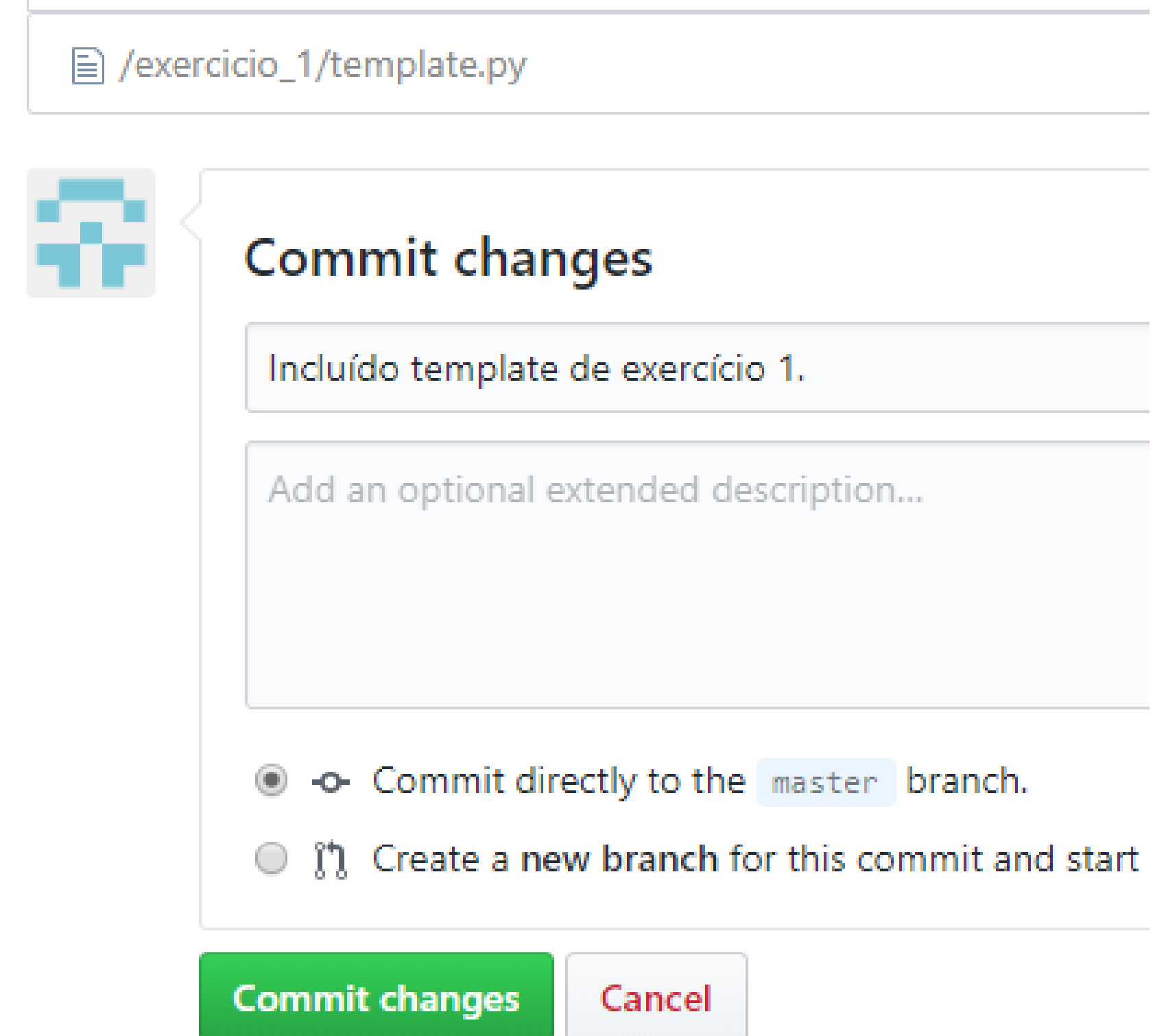
# Algumas tarefas no GitHub

- **Baixando manualmente um repositório para a máquina:**
  - Na tela do seu repositório, clique em **Clone or download**;
  - Em seguida, clique em **Download ZIP**;
  - Com o arquivo .zip baixado, descompacte-o na sua pasta de projeto, substituindo arquivos antigos se necessário.



# Algumas tarefas no GitHub

- **Fazendo um novo commit no seu repositório:**
  - Commits são alterações feitas no repositório. Podem conter um único arquivo ou vários. Caso um arquivo commitado já exista, o GitHub vai fazer um controle de versão, comparando as alterações entre a versão anterior e a que foi commitada;
  - Na tela do seu repositório, clique em **Upload files**;
  - Arraste para a tela os arquivos que deseja incluir;
  - Insira uma breve descrição do que está sendo commitado;
  - Deixe marcada a opção **Commit directly to the master branch**;
  - Clique em **Commit changes**.



The screenshot shows the GitHub 'Commit changes' interface. At the top, a file path `/exercicio_1/template.py` is displayed. Below it is a blue icon representing a commit. The main section is titled 'Commit changes' and contains a text box with the message 'Incluído template de exercício 1.' Below this is a larger text box with the placeholder 'Add an optional extended description...'. At the bottom, there are two radio button options: 'Commit directly to the master branch.' (which is selected) and 'Create a new branch for this commit and start'. At the very bottom are two buttons: 'Commit changes' (green) and 'Cancel' (grey).

# Algumas tarefas no GitHub

- **Submetendo um trabalho para revisão:**
  - Um **pull request** é o ato de submeter para aprovação as alterações ou inserções de código de um ou mais arquivos. A pessoa que abre um **pull request** sinaliza que gostaria de uma aprovação do conteúdo antes de ele ser, de fato, incorporado ao repositório;
  - No repositório desejado, clique na pasta em que você deseja incluir ou atualizar os arquivos;
  - Clique em **Upload files**;
  - Arraste para a tela o(s) arquivo(s) com a sua atualização, e na descrição explique o que está sendo feito. Em seguida, clique em **Commit changes**;
  - Na nova janela, insira um comentário e depois clique em **Create pull request**.



# Alguns comandos do Git

- Apesar do Github fornecer recursos para operar com Git direto pelo navegador, esses recursos são limitados. Por exemplo, fazer checkout de um novo branch apenas pelo navegador pode ser bem complicado.
- Uma forma mais usual de se usar o Git é através de programas específicos para o computador.
- É bem comum utilizar os comandos do Git por linha de comando, porém existem bons programas para uso do Git através de uma interface gráfica, como o [Sourcetree](#).
- No slide a seguir serão apresentados alguns comandos comuns para o uso do Git pela linha de comando. Eles podem ser executados pelo Git Bash (programa instalado junto com o Git), ou pelo terminal.



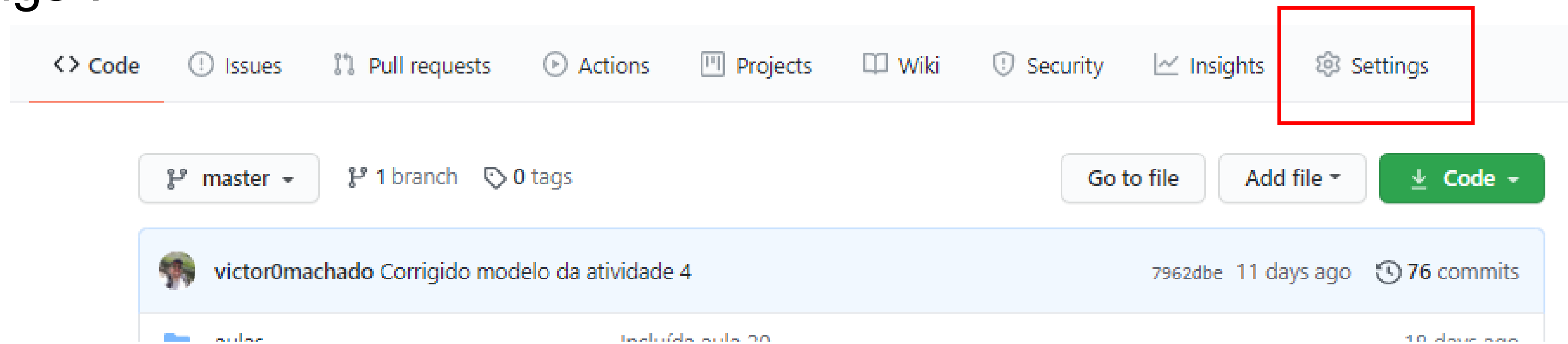
# Alguns comandos do Git

- Uma observação, para facilitar as operações, é trabalhar sempre na raiz do repositório.
- Com o terminal na raiz do repositório, insira os seguintes comandos para obter os efeitos apresentados:

Comando	Efeito
git init	Inicia um repositório Git no diretório em questão
git status	Indica os status de arquivos modificados, adicionados ou removidos, além de arquivos preparados (staged)
git add <arquivo>	Prepara o arquivo mencionado
git add -u	Prepara todos os arquivos modificados (porém não faz nada com arquivos novos)
git add .	Prepara todos os arquivos (incluindo arquivos novos)
git commit -m "Mensagem"	Faz um commit dos arquivos preparados, incluindo a mensagem de commit definida
git restore <arquivo>	Desfaz modificações do arquivo que não foi preparado
git restore --staged <arquivo>	Desfaz a preparação do arquivo (porém mantém modificações)
git pull	No branch escolhido, atualiza as informações com o repositório remoto
git branch	Lista todos os branches armazenados localmente
git branch --show-current	Lista o branch atual
git branch -m <novo_nome>	Renomeia o branch atual (cuidado ao fazer isso para branches que já estão no repositório remoto!)
git checkout <branch>	Dá checkout no branch mencionado
git checkout -b <branch>	Cria um novo branch, com o nome mencionado, e dá checkout nele
git push	Envia os commits realizados localmente para o branch remoto

# GitHub pages

- Uma boa forma de manter o seu portfólio sempre atualizado é com uma página pessoal, na qual você inclui seu currículo, projetos realizados, trabalhos, interesses pessoais e profissionais, e outras informações que achar pertinente.
- O Github possui uma forma muito simples de se criar um repositório que também serve como página pessoal.
- Para isso, crie um repositório normal, vá na página desse repositório e clique em “Settings”.



# GitHub pages

- Nas configurações, desça a página até encontrar a seção “GitHub Pages”.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

**Source**  
GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more.](#)

**Theme Chooser**  
Select a theme to publish your site with a Jekyll theme using the gh-pages branch. [Learn more.](#)

- No campo “Source”, indique o branch que você quer que seja a sua página principal (usualmente é o branch master). Se quiser, escolha um tema da lista de temas gratuitos disponíveis e, em seguida, clique em “Save”.

# GitHub pages

- A página terá uma atualização que mostrará a URL do site, além de incluir um campo no qual você pode inserir um domínio customizado, caso o tenha.

## GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is ready to be published at <https://victor0machado.github.io/2020.2-logprog/>.

### Source

Your GitHub Pages site is currently being built from the master branch. [Learn more.](#)

Branch: master ▾

/ (root) ▾

Save

### Theme Chooser

Select a theme to publish your site with a Jekyll theme. [Learn more.](#)

Choose a theme

### Custom domain

Custom domains allow you to serve your site from a domain other than victor0machado.github.io. [Learn more.](#)

Save

# GitHub pages

- O site funciona como um repositório normal. Todas as páginas devem ser escritas em Markdown, que já vimos ao longo do curso.

```
# Boas-vindas

Vou atualizar essa página com os materiais das disciplinas que leciono no
IBMEC/RJ.

## Disciplinas

* [Algoritmos e Programação de Computadores](/courses/algprog.md)
* [Lógica e Programação de Computadores](/courses/logprog.md)
* [Data Mining com Python](/courses/datamining.md)

## Meus contatos

* E-mail: <victor.silva@professores.ibmec.edu.br>
* [LinkedIn](https://www.linkedin.com/in/victormachadodasilva/)
* [Lattes](http://lattes.cnpq.br/1584907276781609)
```

## Repositório público do Prof. Victor Machado

Material usado nas minhas disciplinas do  
IBMEC/RJ

[View My GitHub Profile](#)

## Boas-vindas

Vou atualizar essa página com os materiais das disciplinas que leciono no  
IBMEC/RJ.

## Disciplinas

- Algoritmos e Programação de Computadores
- Lógica e Programação de Computadores
- Data Mining com Python

## Meus contatos

- E-mail: victor.silva@professores.ibmec.edu.br
- LinkedIn
- Lattes

- O único arquivo exigido para o site é o **index.md**, que deve ficar na raiz do repositório. Novos arquivos e pastas podem ser criados se necessário, e a navegação é sempre relativa à raiz do repositório.

OBRIGADO!



[www.ibmec.br](http://www.ibmec.br)

 /ibmec

 ibmec

 @ibmec\_oficial

 ibmec

