

Aprendendo



Bismarck Gomes Souza Júnior

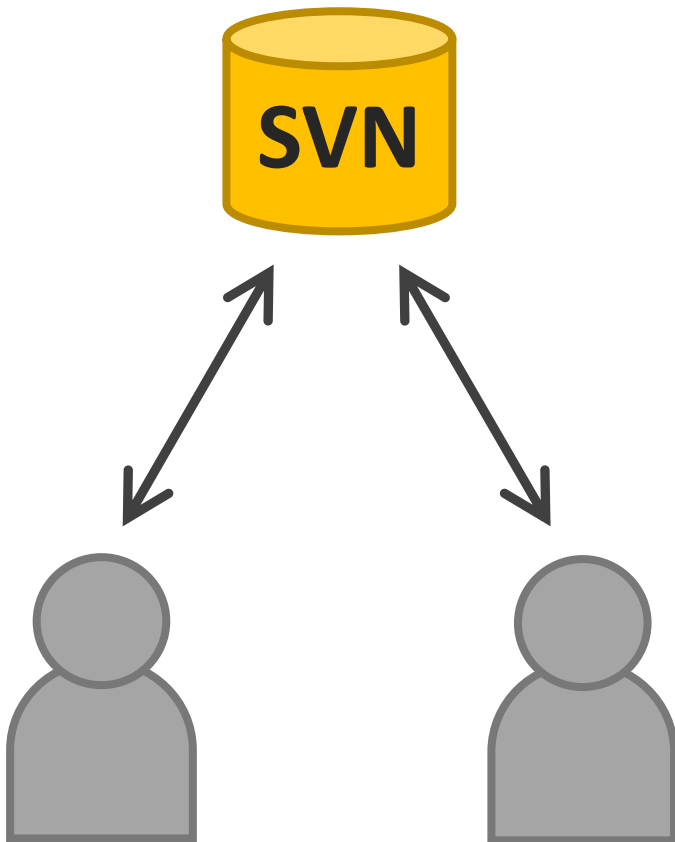
Março de 2014



Sistema de Controle de Versão

- Controle de histórico
- Trabalho em equipe
- Marcação e resgate de versões estáveis
- Ramificação do Projeto

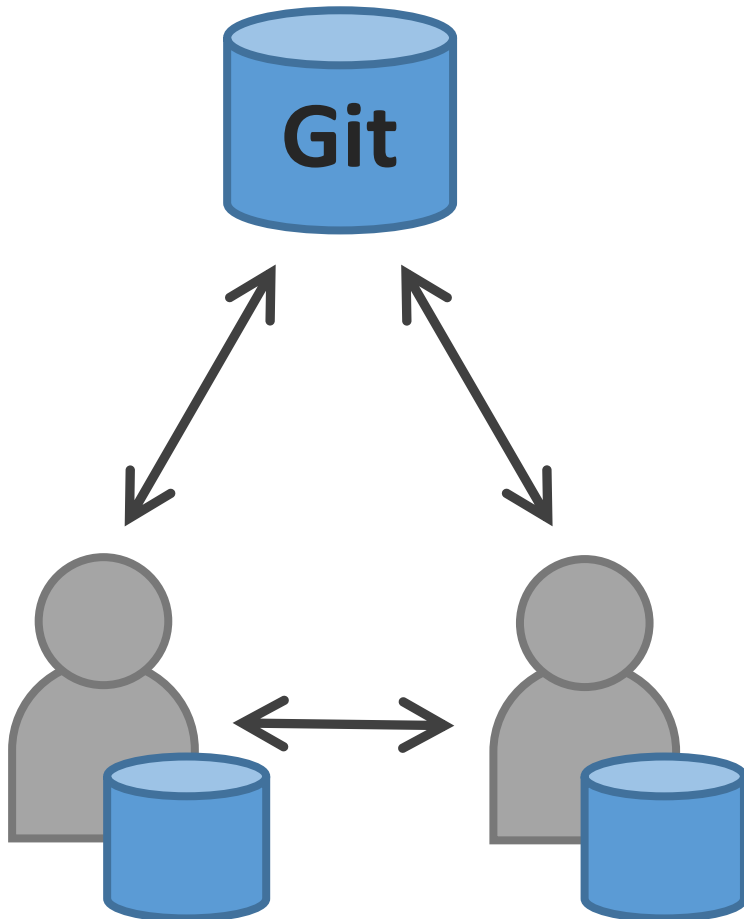
Sistema de Controle de Versão Centralizado



- Pouca autonomia
Ações necessitam de acesso ao servidor.
- Trabalho privado limitado
Versiona apenas arquivos no repositório.
- Risco de perda de dados
Tudo em um único repositório.



Sistema de Controle de Versão Distribuído



- **Autonomia**
Ações básicas “*off-line*”.
- **Rapidez**
Processos são locais.
- **Trabalho privado**
Trabalho local não afeta os demais.
- **Confiabilidade**
Todo repositório é um *backup*, ou seja, uma cópia completa do repositório, incluindo versões anteriores e histórico.



Sistema de Controle de Versão



Início



commit



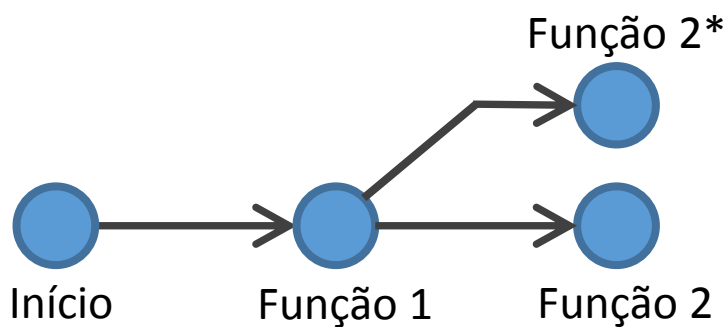
Sistema de Controle de Versão



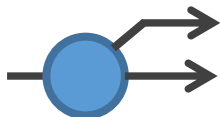
 **commit**



Sistema de Controle de Versão



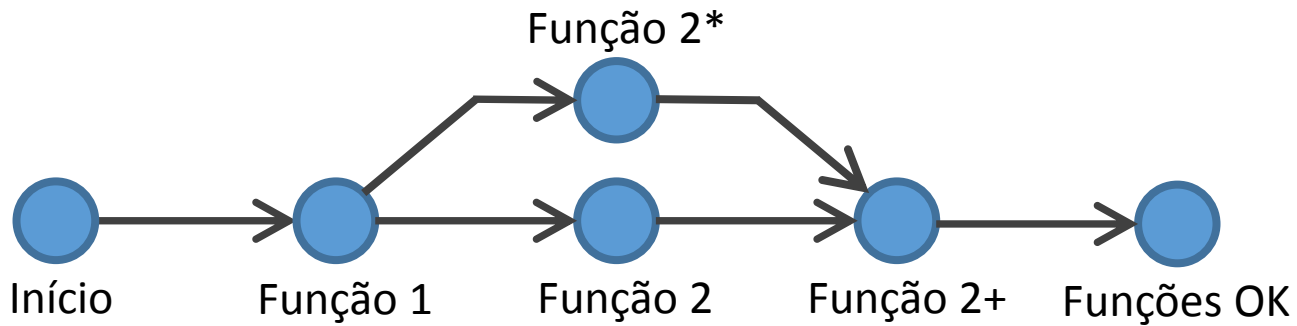
commit



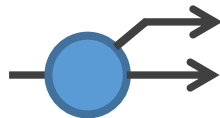
branch

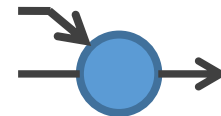


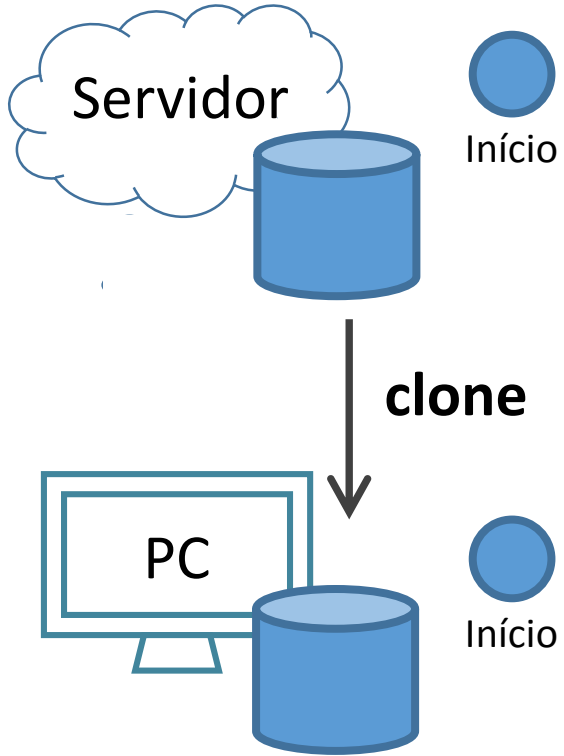
Sistema de Controle de Versão

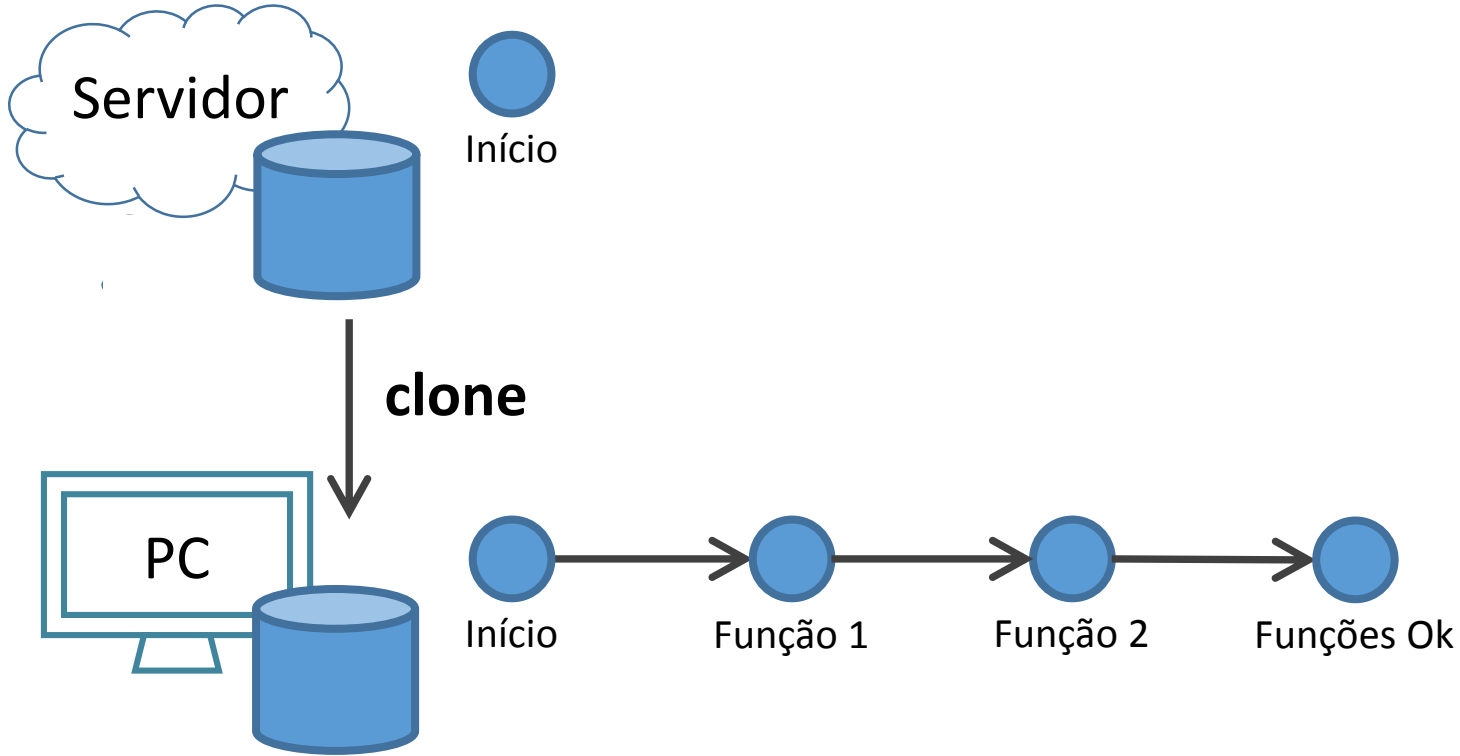


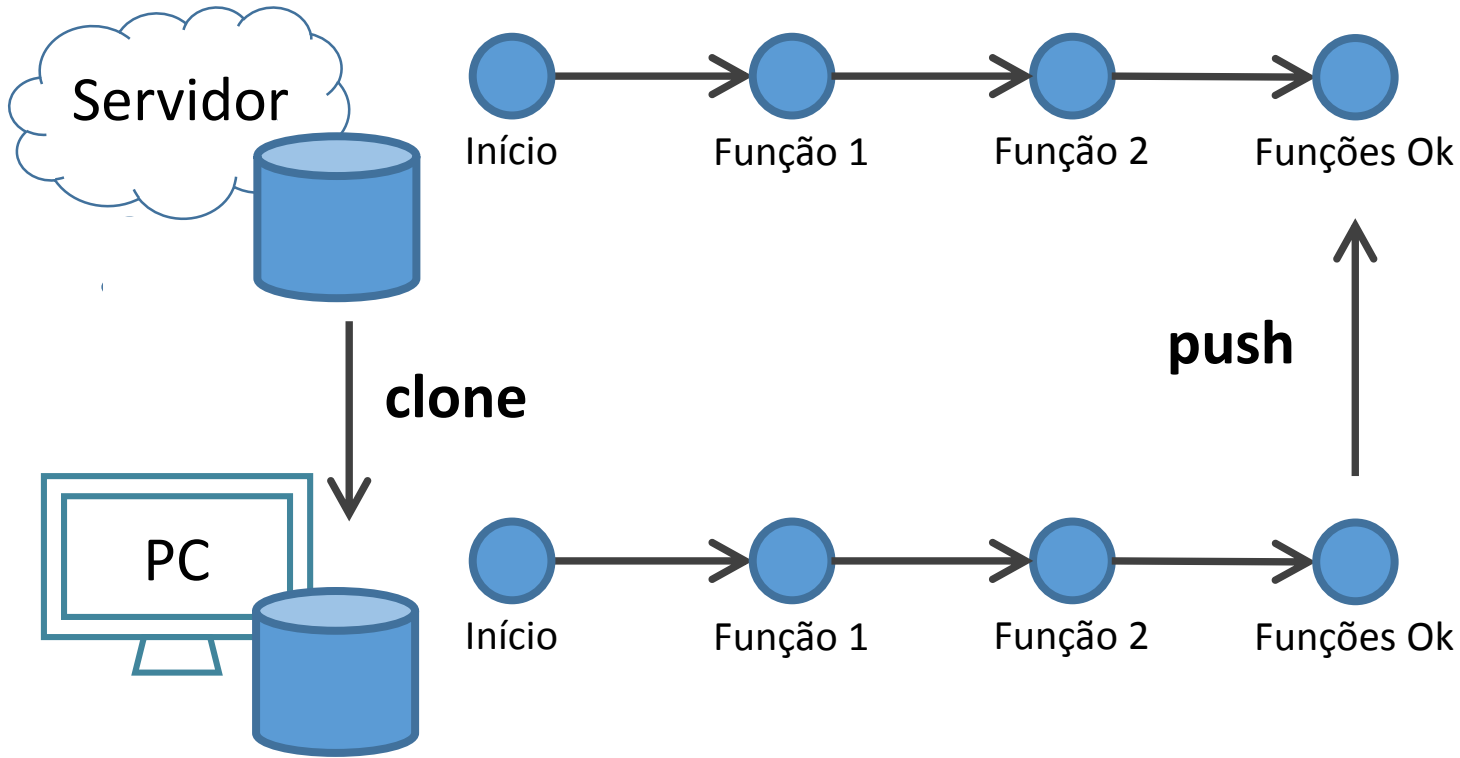
 **commit**

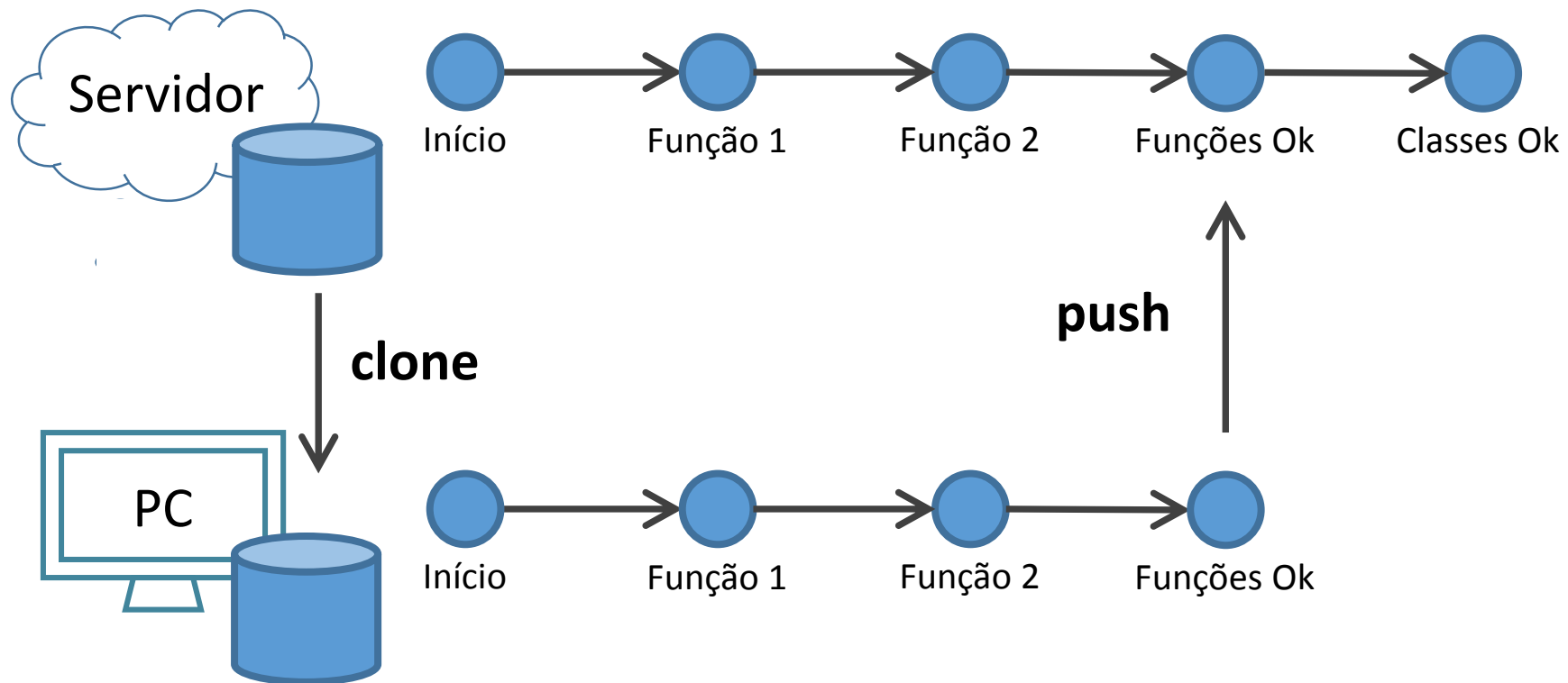
 **branch**

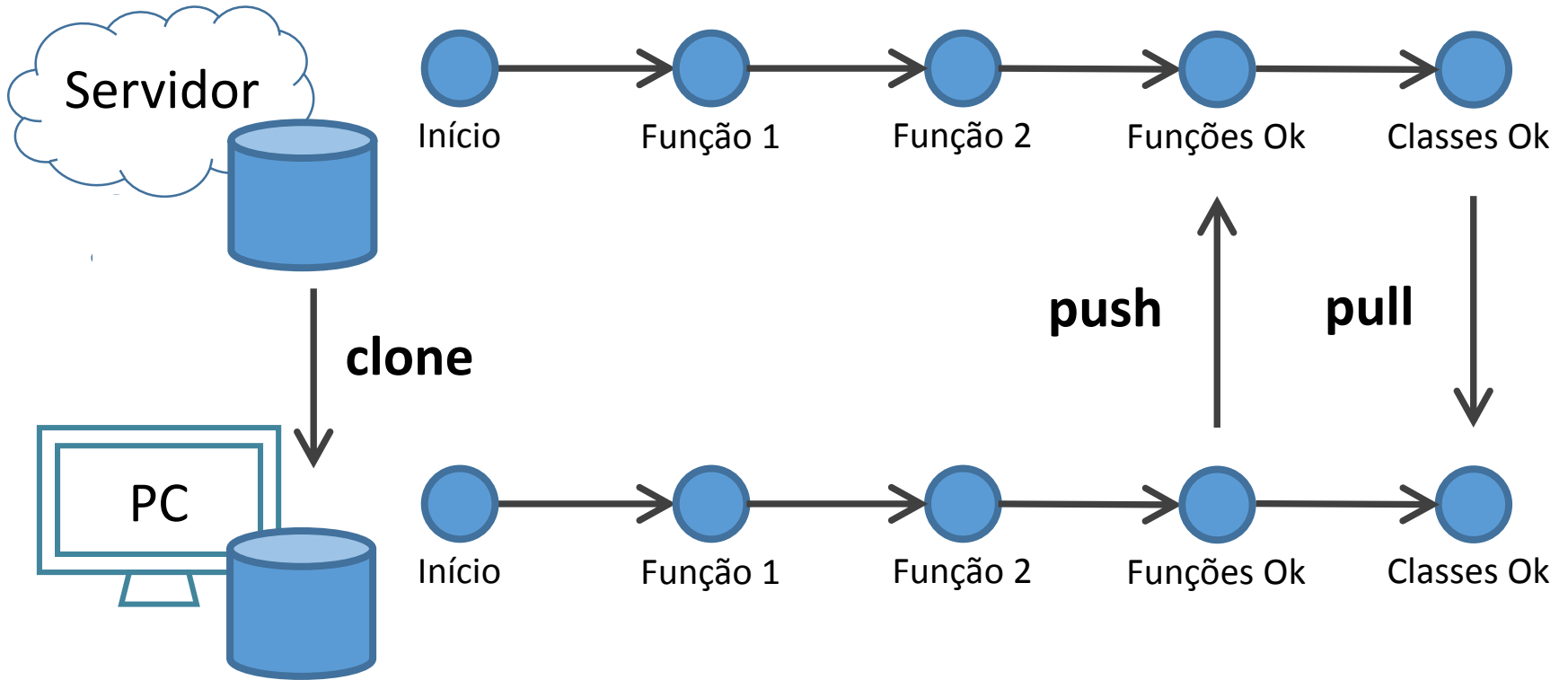
 **merge**













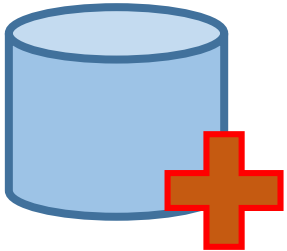
Servidores Para Hospedagem



Comandos Básicos



Criando um Repositório



```
$ git init
```

Transforma a diretório atual em um repositório git, criando o subdiretório “.git”.

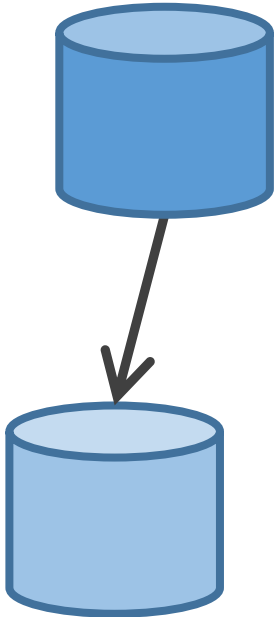


```
$ git init <dir>
```

Cria o diretório <dir> e transforma em um repositório git.



Clonando um Repositório



```
$ git clone <repo>
```

Clona o repositório <repo> para a máquina local.

```
$ git clone <repo> <dir>
```

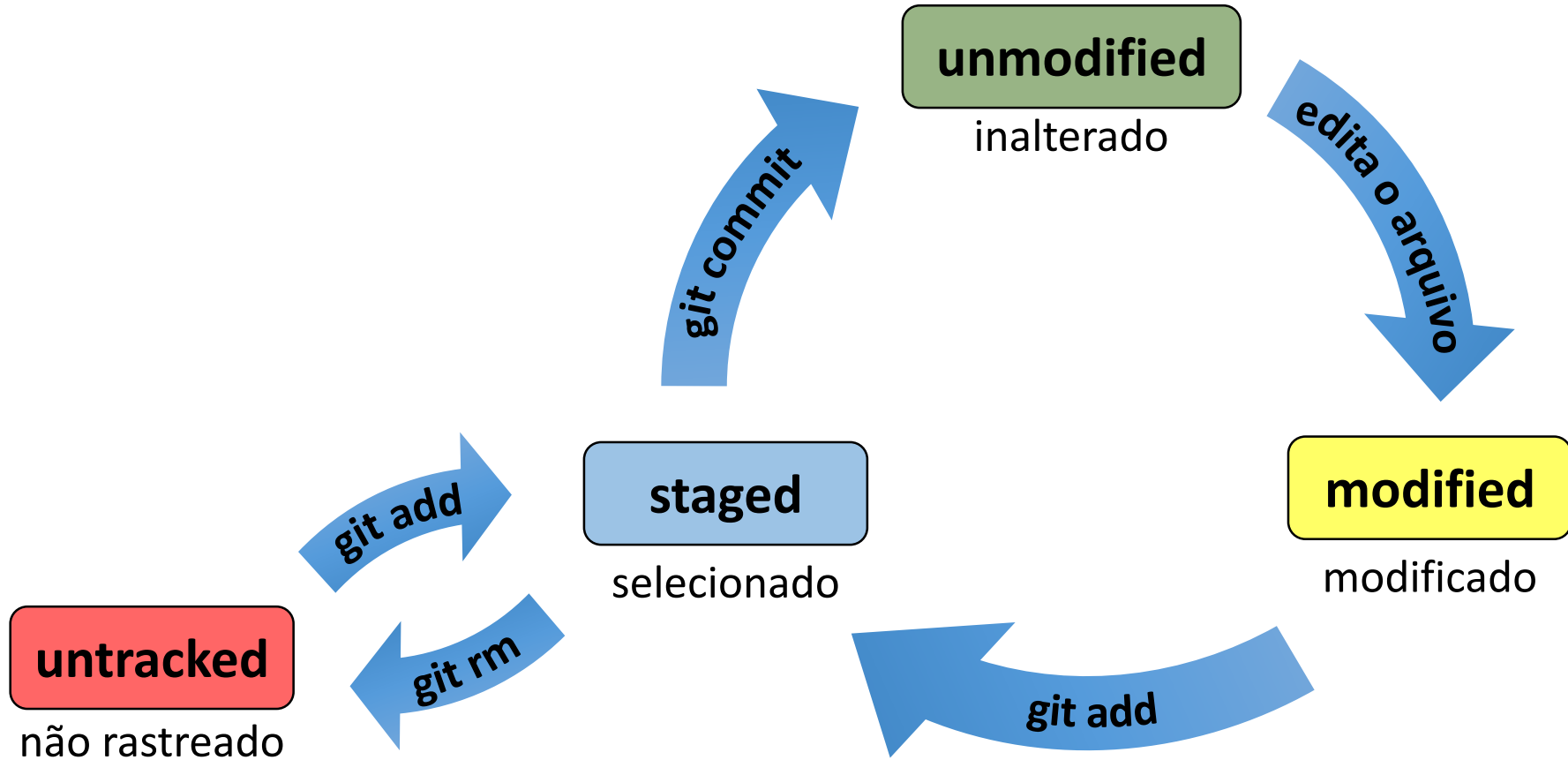
Clona o repositório <repo> para o diretório <dir>.

```
$ git clone git@github.com:user/Project.git
```

```
$ git clone https://github.com/user/Project.git
```



Tipos de Estado de um Arquivo





.gitignore

Arquivo que contém os arquivos que não serão visíveis pelo git.

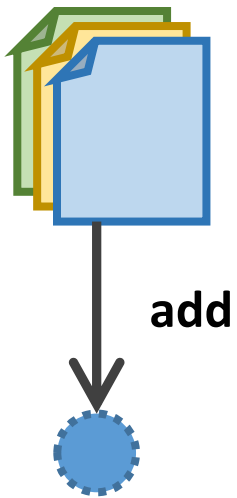
Arquivo .gitignore (exemplo)

```
Thumbs.db      #Arquivo específico
*.html         #Arquivos que terminam com “.html”
!index.html    #Exceção, esse arquivo será visível ao git
log/           #Diretório específico
**/tmp        #Qualquer diretório nomeado de “tmp”
```

- Arquivos que já estavam sendo rastreados não são afetados.



Preparando Para Salvar Alterações



Stage Area
(Index)

```
$ git add <arquivo|dir>
```

Adiciona as mudanças do arquivo <arquivo> ou do diretório <dir> para o próximo *commit*. O arquivo passa a ser rastreado.

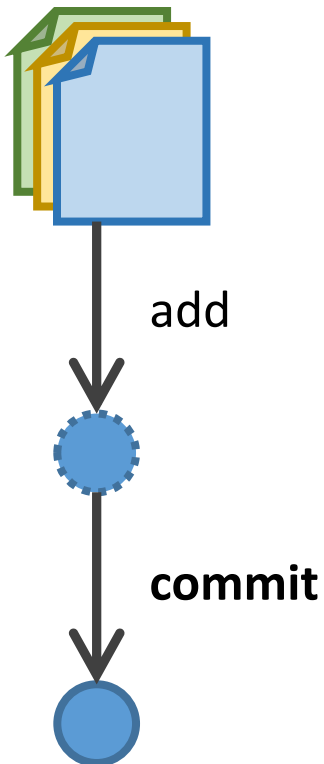
```
$ git reset <arquivo>
```

Remove as mudanças do arquivo <arquivo> para o próximo *commit*.

```
$ git rm --cached <arquivo>
```

Para de rastrear o arquivo <arquivo>.

Salvando Alterações



```
$ git commit
```

Realiza o *commit* e abre o editor para inserir uma mensagem.

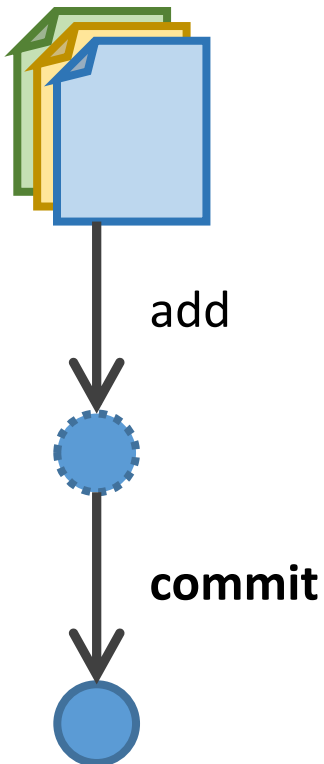
```
$ git commit -a
```

Adiciona as mudanças dos arquivos já rastreados e realiza o *commit*. O editor será aberto.

```
$ git commit -m "<msg>"
```

Realiza o *commit*, com a mensagem <msg>.

Salvando Alterações



```
$ git commit -am <msg>
```

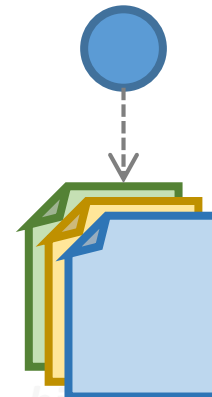
Adiciona as mudanças dos arquivos já rastreados e realiza o *commit* com a mensagem <msg>.

```
$ git commit --amend -m <msg>
```

Substitui o último commit e altera a mensagem para <msg>.



Commmmit





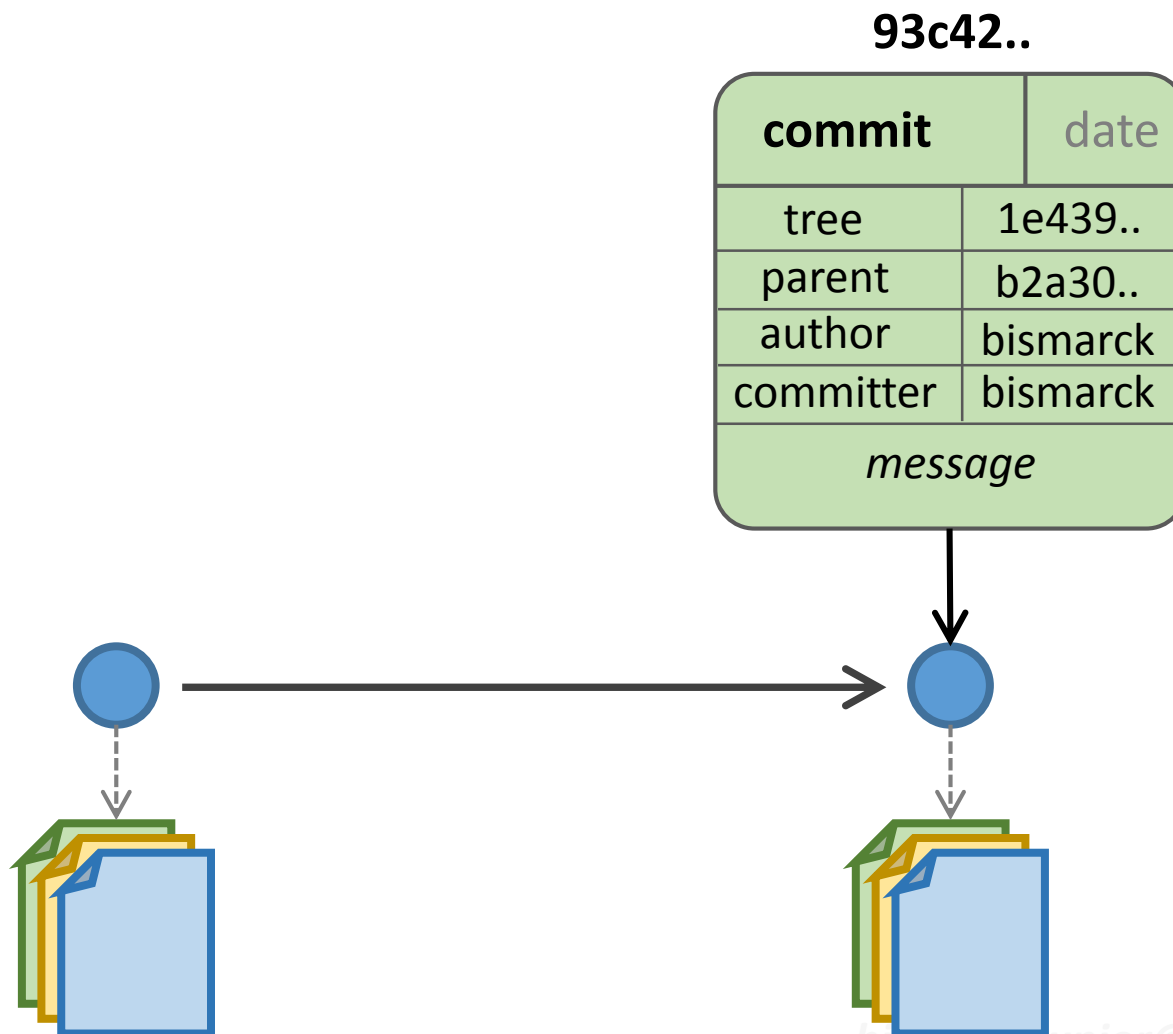
Commmmit

93c42..

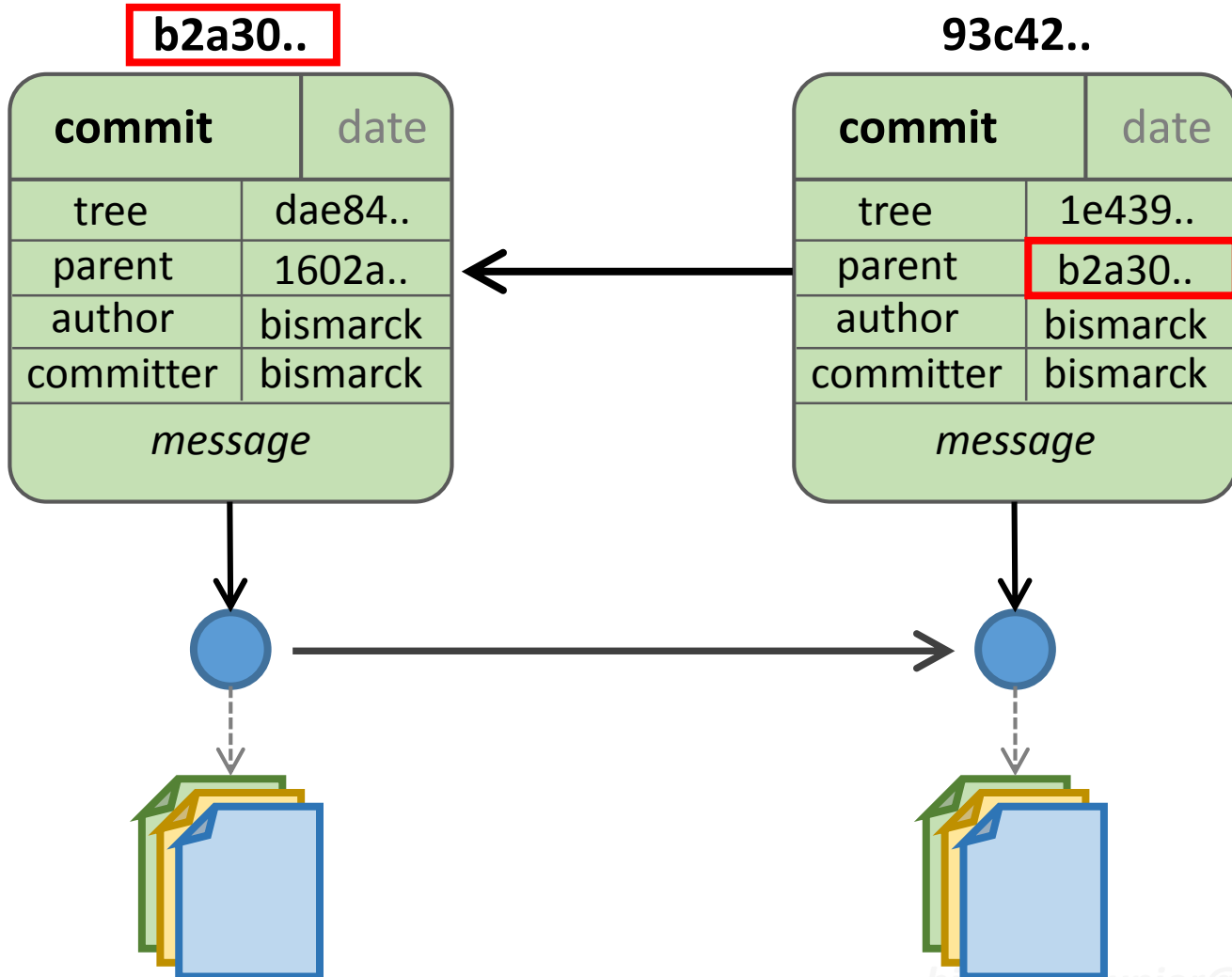
commit	date
tree	1e439..
parent	b2a30..
author	bismarck
committer	bismarck
<i>message</i>	



Commmmit

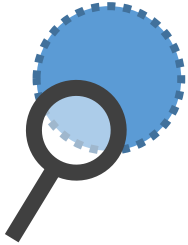


Commmmit





Analizando os Arquivos na Área Transitória



```
$ git status
```

Lista os arquivos que estão e que não estão na área transitória, e os arquivos que não estão sendo rastreados.

```
$ git status -s
```

Lista os arquivos de uma forma simplificada.

Tagging



```
$ git tag
```

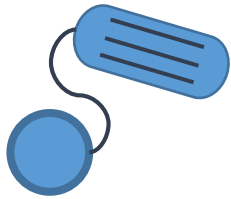
Lista as *tags* existentes.

```
$ git tag -l <tag>
```

Procura pela *tag* <tag>.

```
$ git tag -l 'v.0.*'
```

Tagging



```
$ git tag <tag> [<commit>]
```

Cria a *tag* <tag> para o último *commit* ou para o *commit* <commit>.

```
$ git tag -a <tag>
```

Cria a *tag* <tag> completa para o último *commit* e abre o editor para inserir uma mensagem.

```
$ git tag -a <tag> -m <msg>
```

Cria a *tag* <tag> completa para o último *commit* com a mensagem <msg>.





Versionamento



v.0.1.0

v [major] . [minor] . [patch]

[patch]: correção de *bugs*.

[minor]: incrementos de funcionalidades compatíveis com versões anteriores.

[major]: incrementos de funcionalidades incompatíveis com versões anteriores.

Versões teste: alpha (a), beta (b)

Ex: v0.1.9 < v0.1.10 < v0.2.0a < v0.2.0b < v0.2.0



Referência a *Commit*

<sha1>

Hash SHA-1 referente ao *commit*. Pode-se usar os primeiros caracteres.

Ex: b230 = b230e84a4c90d2f11ba85404e5fba93ce0a...

<tag>

Tag referente ao *commit*.

Ex: v0.1.2

<branch>

Último *commit* do *branch* <branch>.

Ex: master

Analizando *Commits*



```
$ git show
```

Exibe o último *commit*.

```
$ git show <commit>
```

Exibe o *commit* referenciado por <commit>.

```
$ git show <commit>:<arquivo>
```

Exibe o arquivo <arquivo> no *commit* <commit>.

Analizando um Arquivo



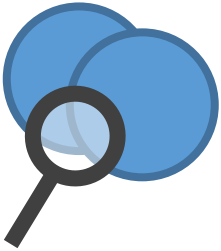
```
$ git blame <arquivo>
```

Exibe quem modificou cada linha do arquivo <arquivo>, incluindo data e *commit*.

```
$ git blame -L <n>,<m> <arquivo>
```

Exibe quem modificou as linhas de <n> a <m> do arquivo <arquivo>, incluindo data e *commit*.

Diferença Entre *Commits*

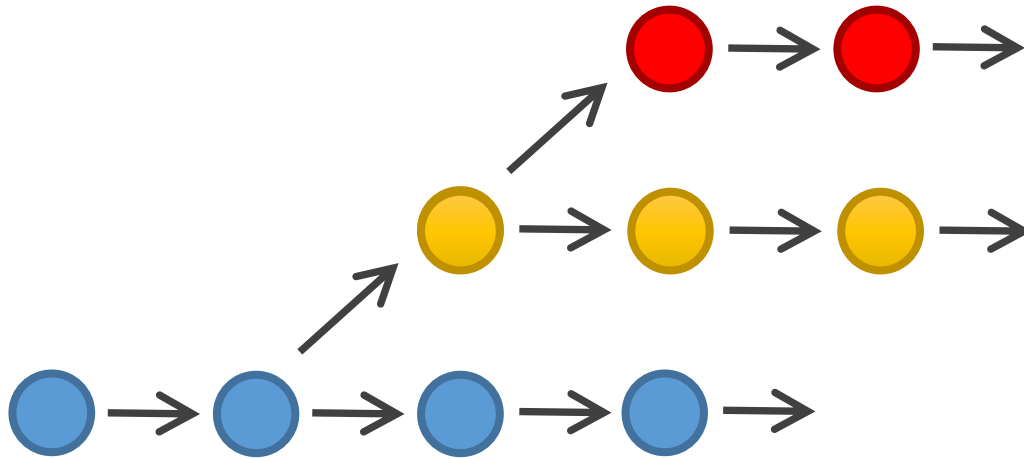


```
$ git diff <commit>
```

Exibe a diferença nos arquivos entre o *commit* <commit> e o diretório de trabalho.

```
$ git diff --cached <commit>
```

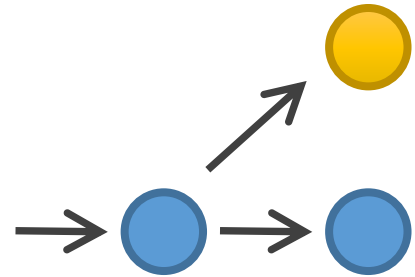
Exibe a diferença nos arquivos entre o *commit* <commit> e a área transitória.



Branches



Criando Ramificações



```
$ git branch [-a]
```

Exibe os *branches* existentes. Na forma completa, exibe também os *branches* remotos.

```
$ git branch <branch> [<base>]
```

Cria o *branch* <branch> a partir do *commit* <base>.

```
$ git checkout -b <branch>
```

Cria o *branch* <branch> e altera para ele.



Criando Ramificações

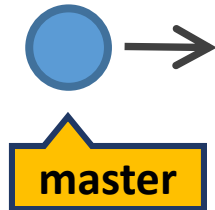


```
$ git add *
```

Adiciona os arquivos para o *index* (área transitória).



Criando Ramificações

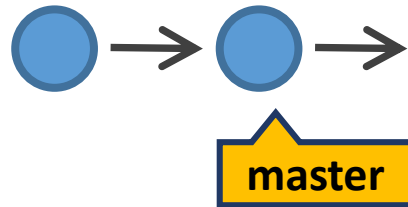


```
$ git commit
```

Realiza um *commit*.



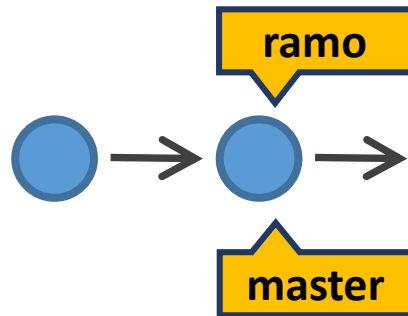
Criando Ramificações



```
$ git commit -a
```

Adiciona os arquivos para o *index* e realiza um *commit*.

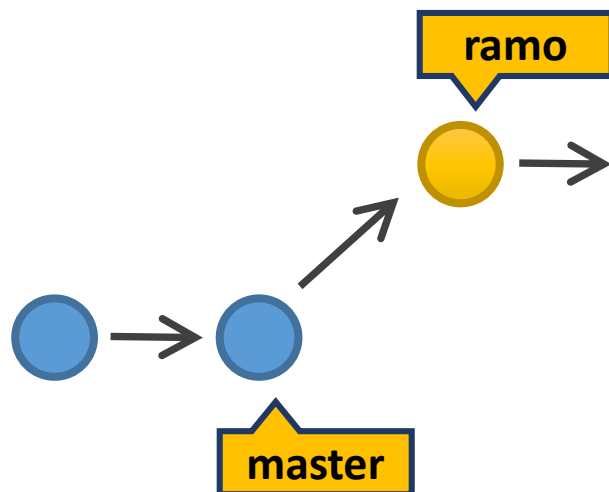
Criando Ramificações



```
$ git checkout -b ramo
```

Cria o *branch* ramo e altera para ele, ou seja, os próximos *commits* serão no *branch* ramo.

Criando Ramificações



```
$ git commit -a
```

Realiza um *commit* no *branch* ramo.



Alternando em Ramificações

```
$ git checkout <branch>
```

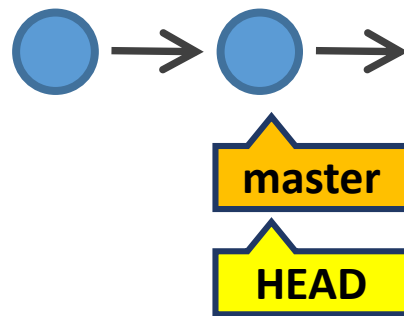
Altera para o *branch* <branch>.

```
$ git checkout -f <branch>
```

Altera para o *branch* <branch> “na força”, perdendo-se as informações não “commitadas”.



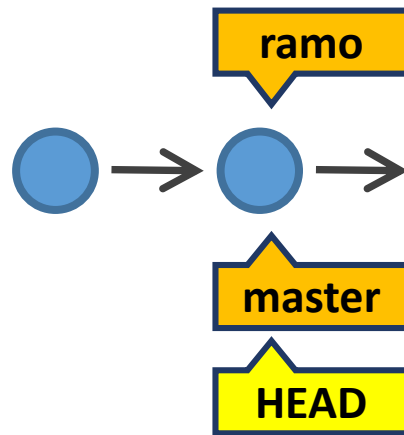
Alternando em Ramificações



HEAD: aponta para o *branch* atual.



Alternando em Ramificações

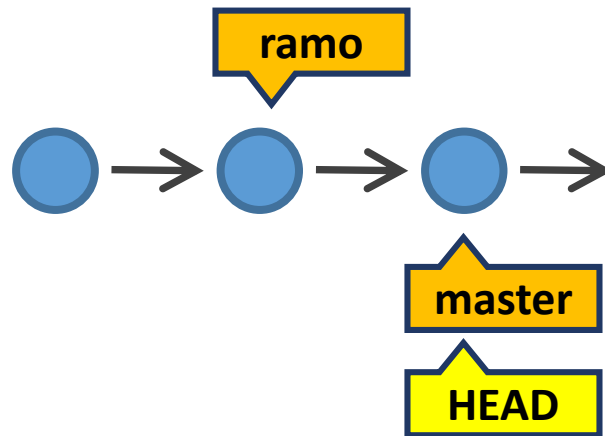


```
$ git branch ramo
```

Cria o *branch* ramo.



Alternando em Ramificações

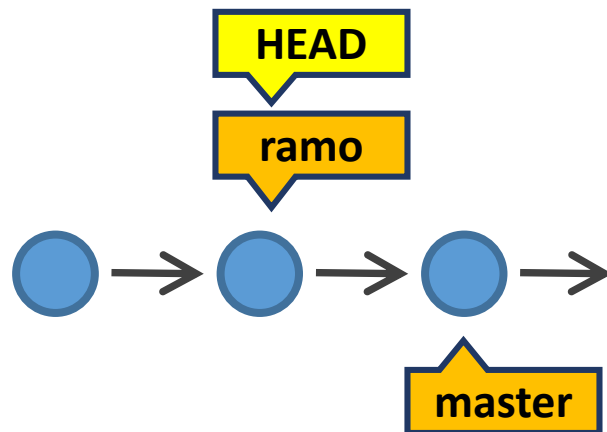


```
$ git commit -a
```

Realiza um *commit* no *branch* master.



Alternando em Ramificações

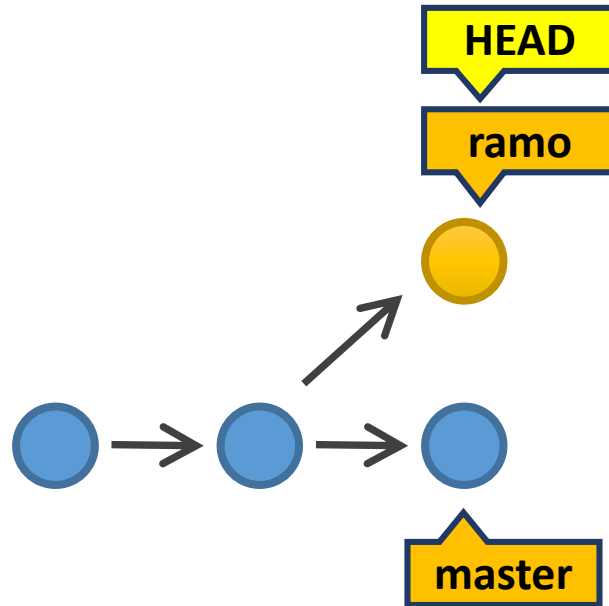


```
$ git checkout ramo
```

Alterna para o *branch* ramo.



Alternando em Ramificações

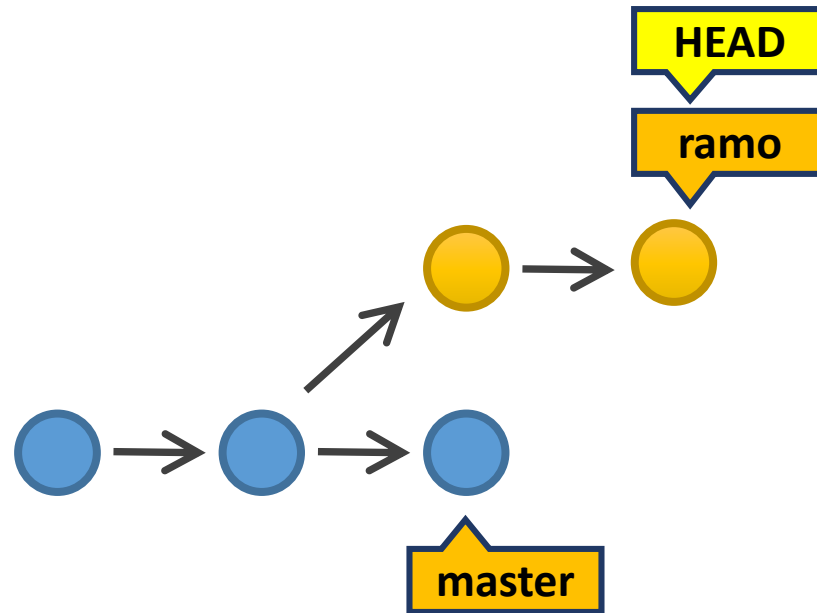


```
$ git commit -a
```

Realiza um *commit* no *branch* ramo.



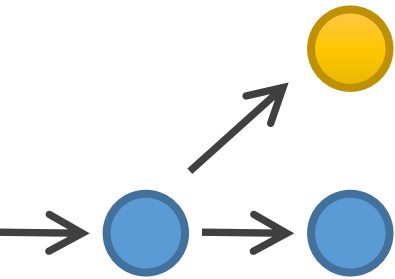
Alternando em Ramificações



```
$ git commit -a
```

Realiza um *commit* no *branch* ramo.

Excluindo Ramificações



```
$ git branch -d <branch>
```

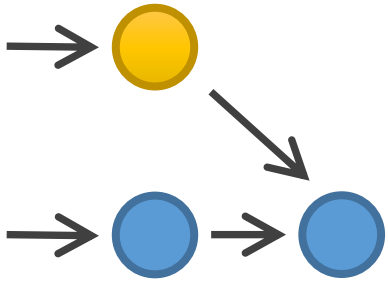
Exclui o *branch* <branch>. O *branch* já deve ter sido mesclado.

```
$ git branch -D <branch>
```

Exclui o *branch* <branch> mesmo não tendo sido mesclado.



Mesclando *Commits*



```
$ git merge <branch>
```

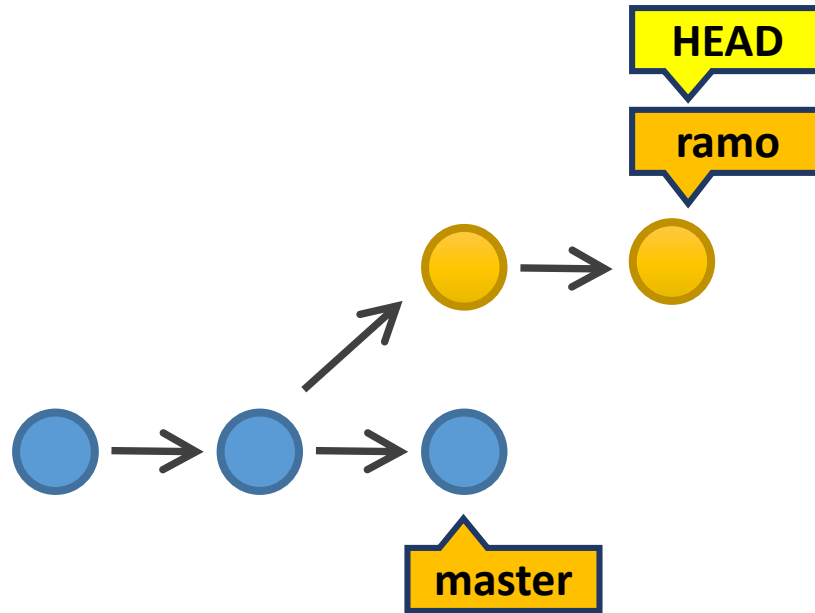
Mescla os *commits* do *branch* <branch> para o branch atual.

```
$ git merge <branch> --no-ff
```

Mescla os *commits* do *branch* <branch> para o branch atual sem *fast-forward*.

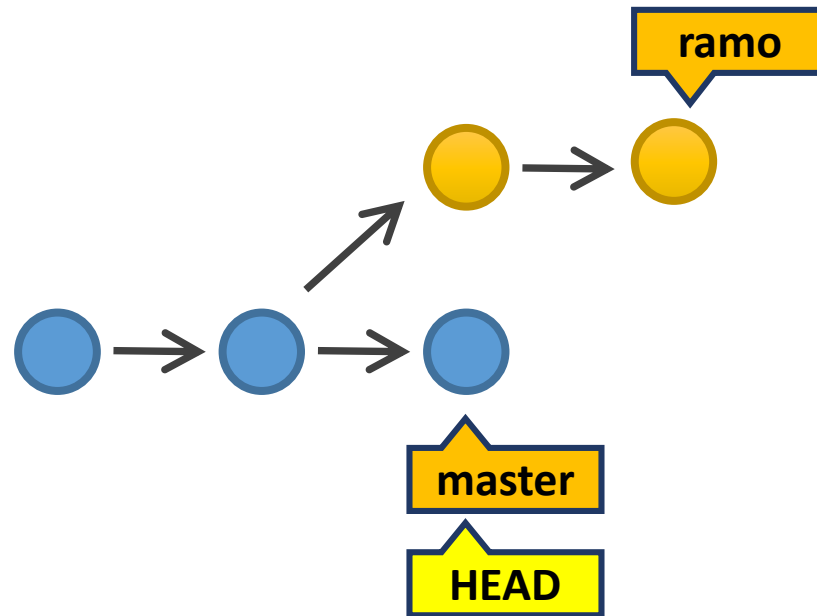


Mesclando *Commits*





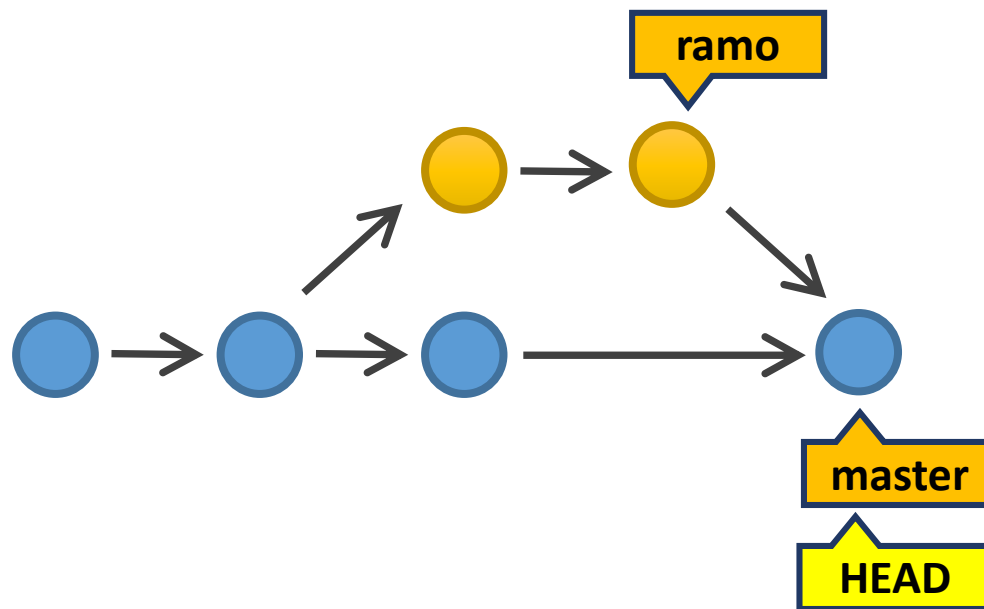
Mesclando *Commits*



```
$ git checkout master
```

Alterna para o *branch* master.

Mesclando *Commits*

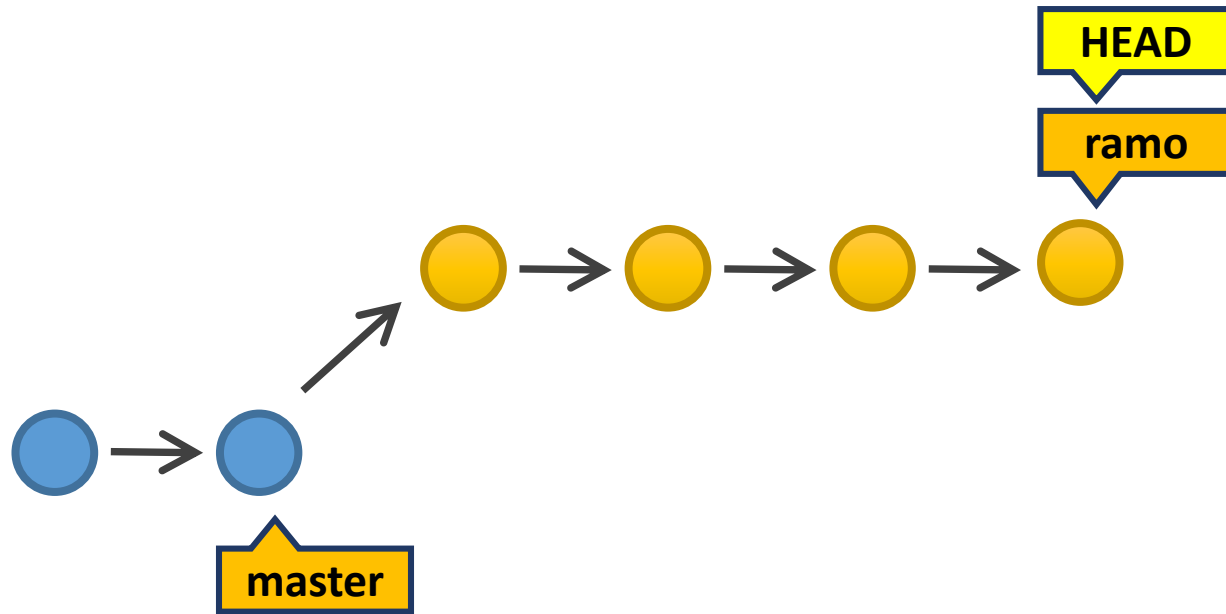


```
$ git merge ramo
```

Realiza um *merge* no *branch* master a partir do *branch* ramo.

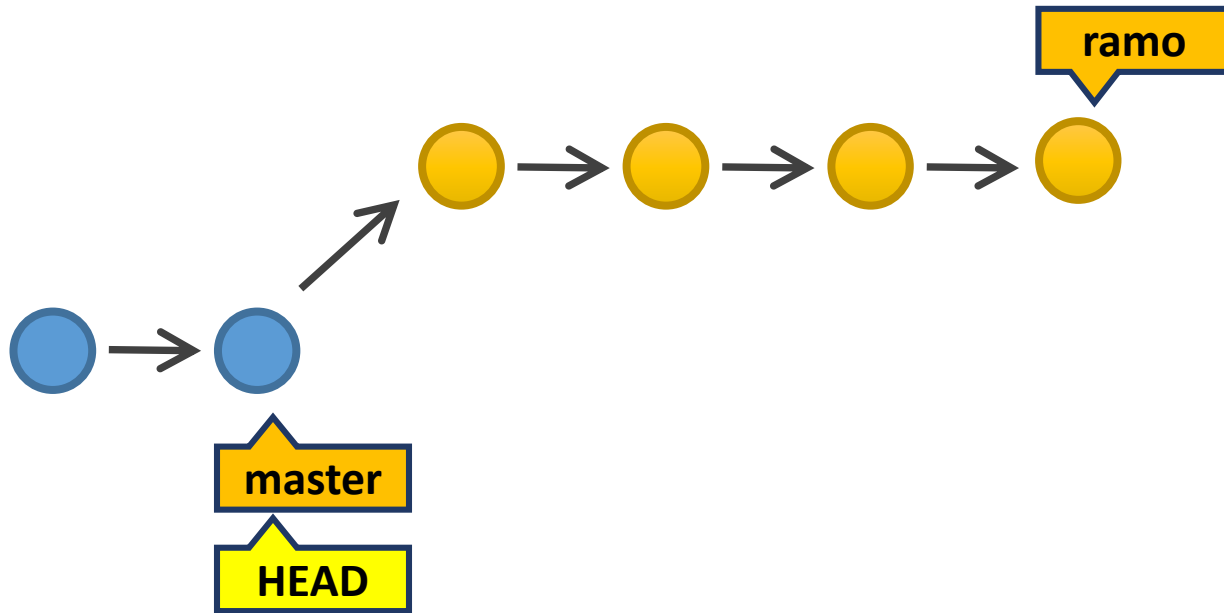


Mesclando *Commits* com *Fast-forward*





Mesclando *Commits* com *Fast-forward*

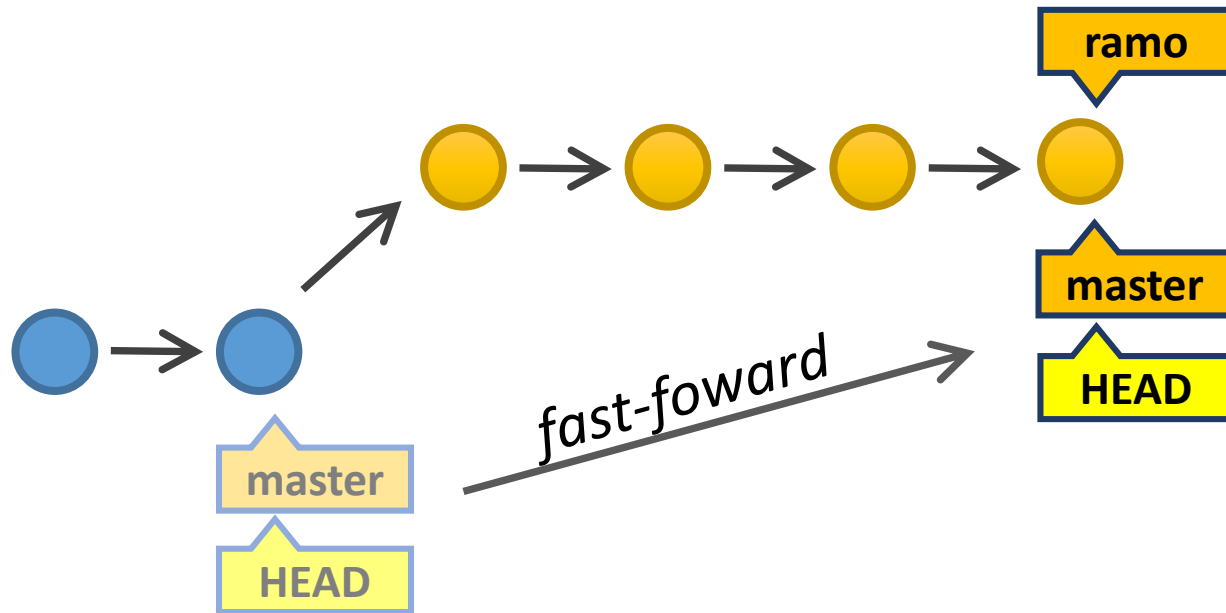


```
$ git checkout master
```

Alterna para o *branch* master.



Mesclando *Commits* com *Fast-forward*

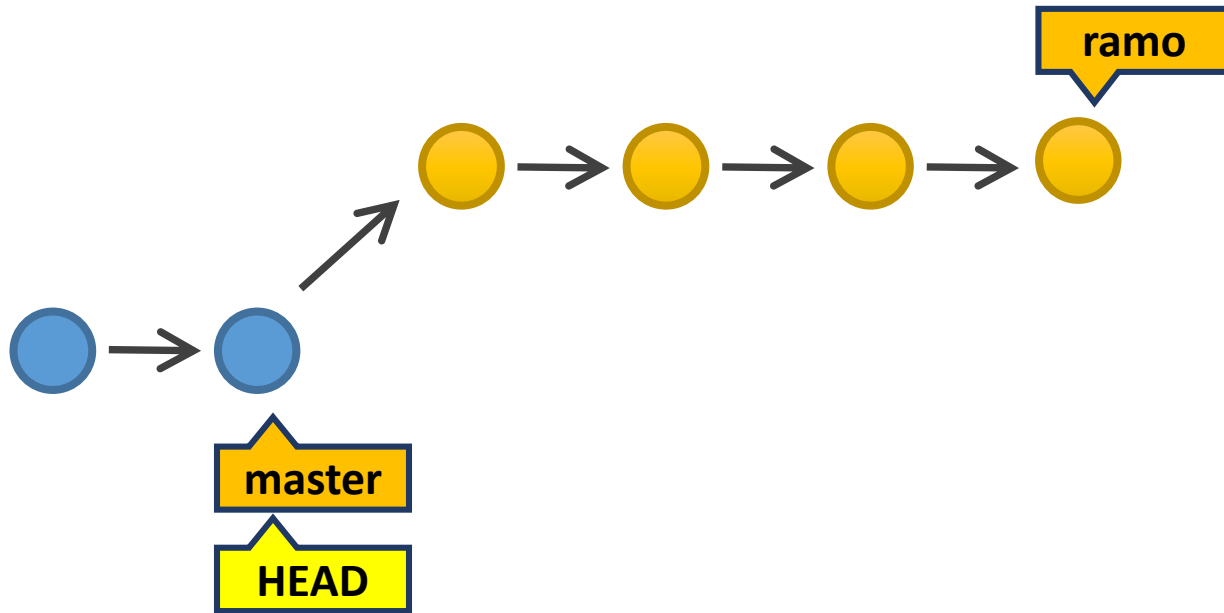


```
$ git merge ramo
```

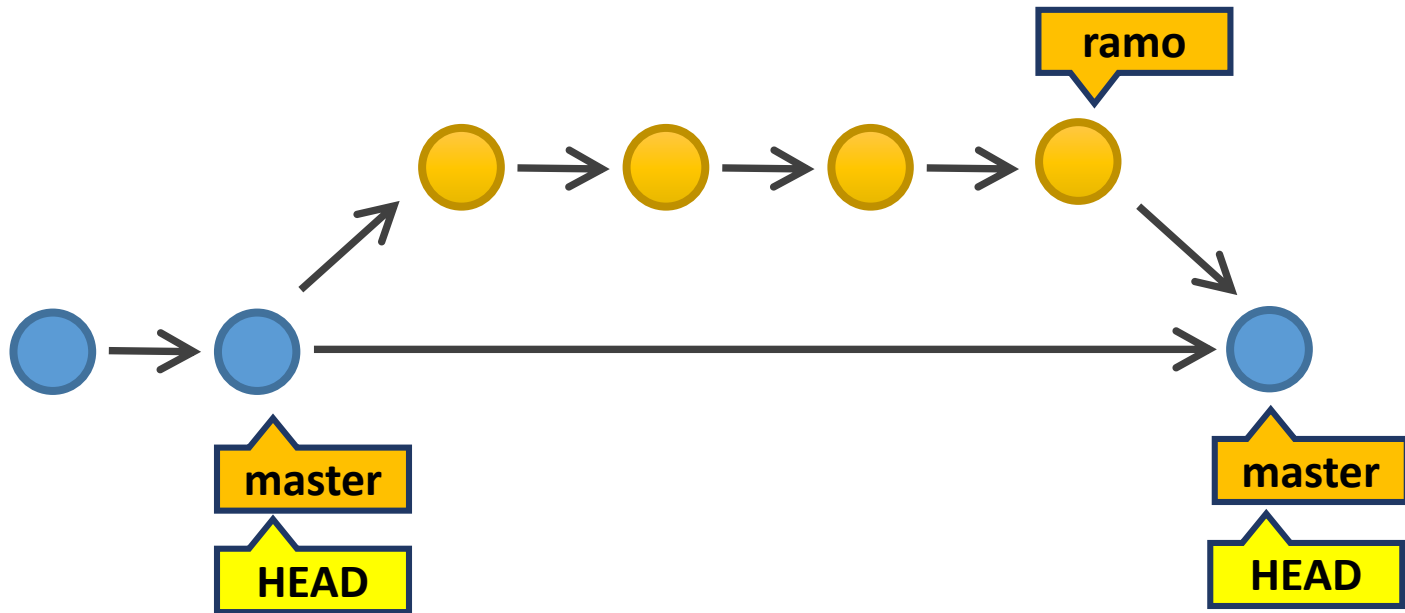
Neste caso, não é necessário nenhum *commit* para realizar a mesclagem. Ocorre apenas um avanço rápido (*ff*).



Mesclando *Commits* sem *Fast-forward*



Mesclando *Commits* sem *Fast-forward*



```
$ git merge ramo --no-ff
```

Realiza um *merge* com um *commit* obrigatoriamente.
Possibilita uma melhor visualização no histórico.



Mesclando *Commits*

master

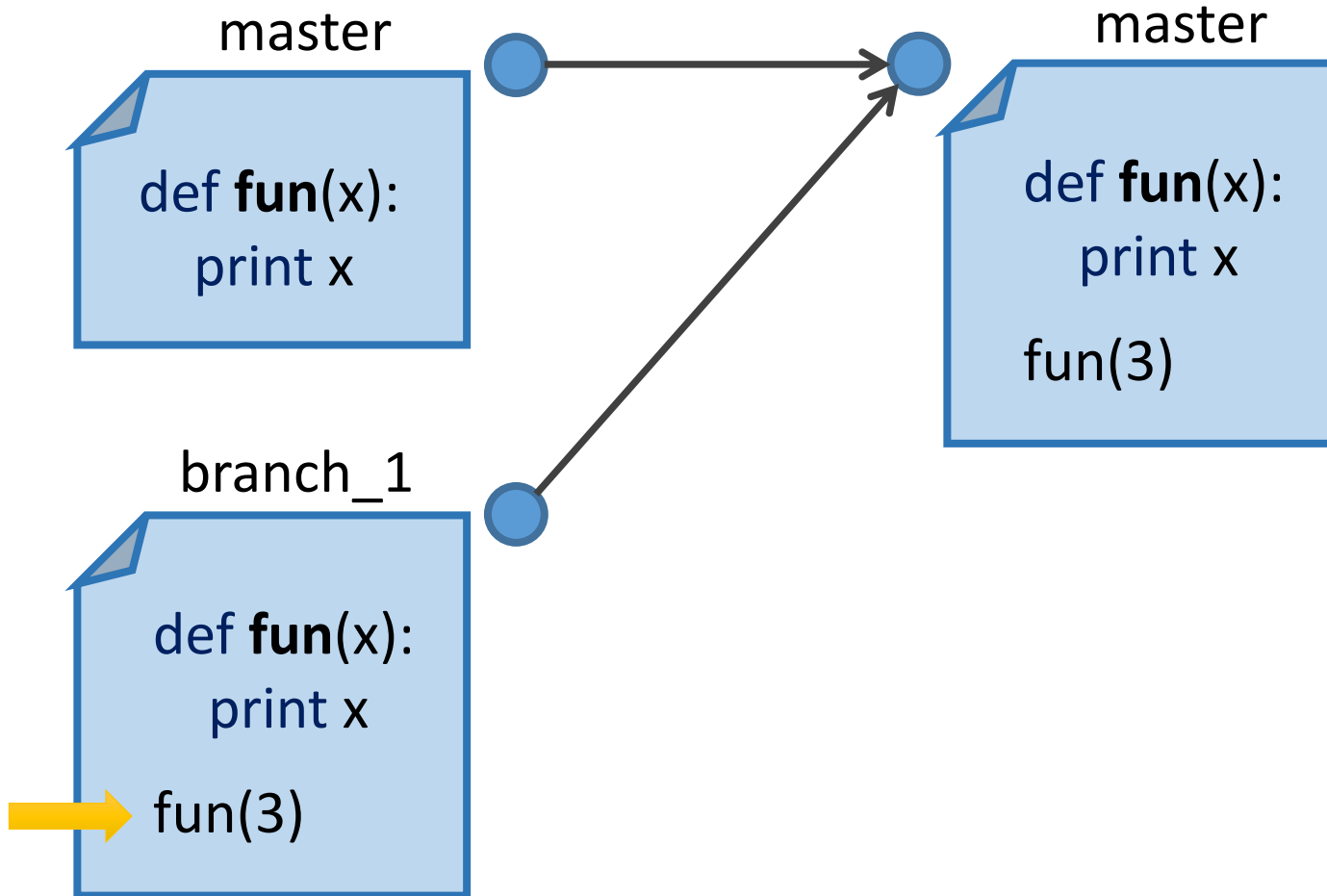
```
def fun(x):  
    print x
```

branch_1

```
def fun(x):  
    print x  
fun(3)
```



Mesclando *Commits*





Mesclando *Commits*

branch_2



```
def fun(x):  
    print x+x
```



branch_1

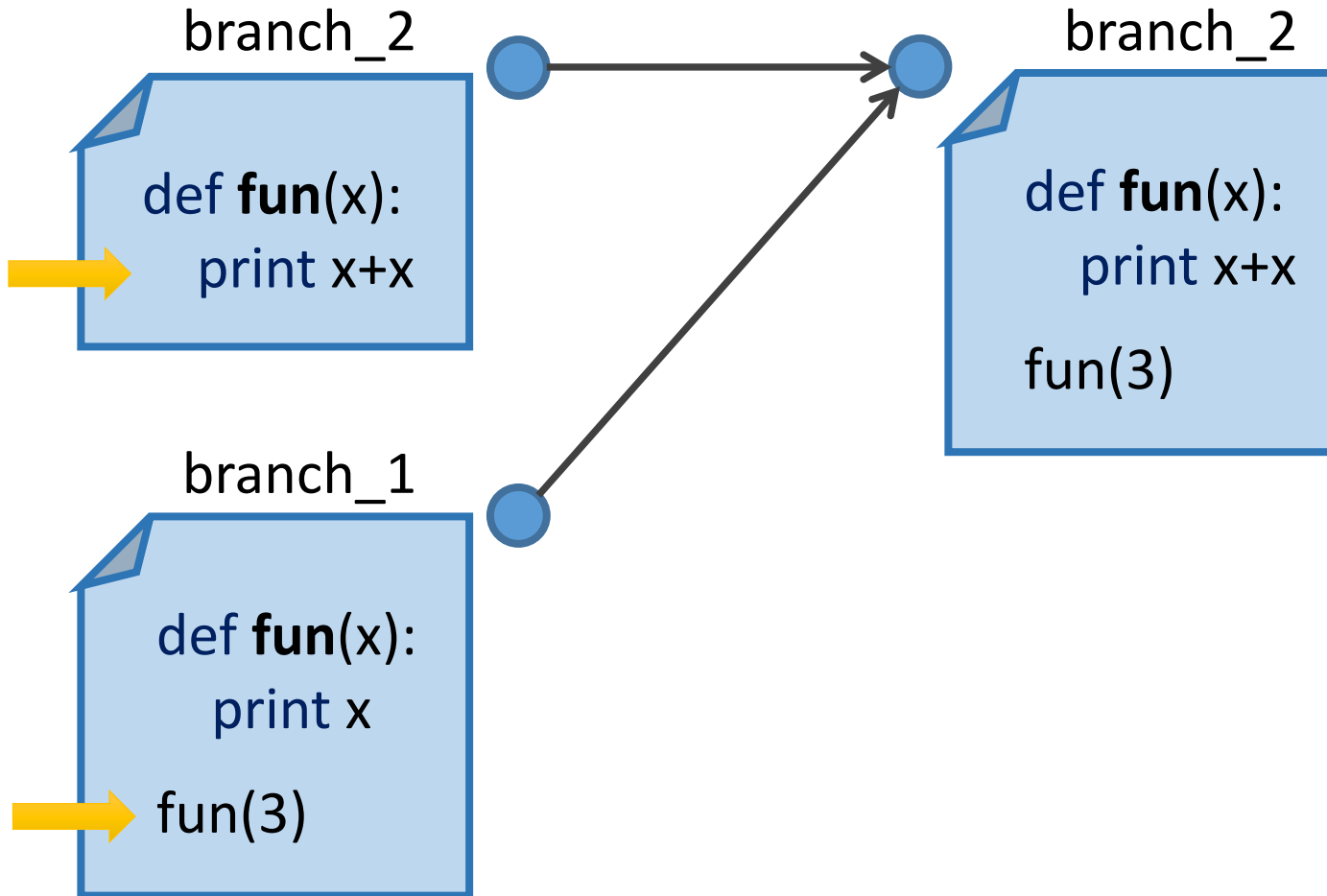


```
def fun(x):  
    print x  
  
fun(3)
```





Mesclando *Commits*





Mesclando *Commits*

branch_2

A thick yellow arrow pointing from the left towards the code block.

```
def fun(x):  
    print x+x
```

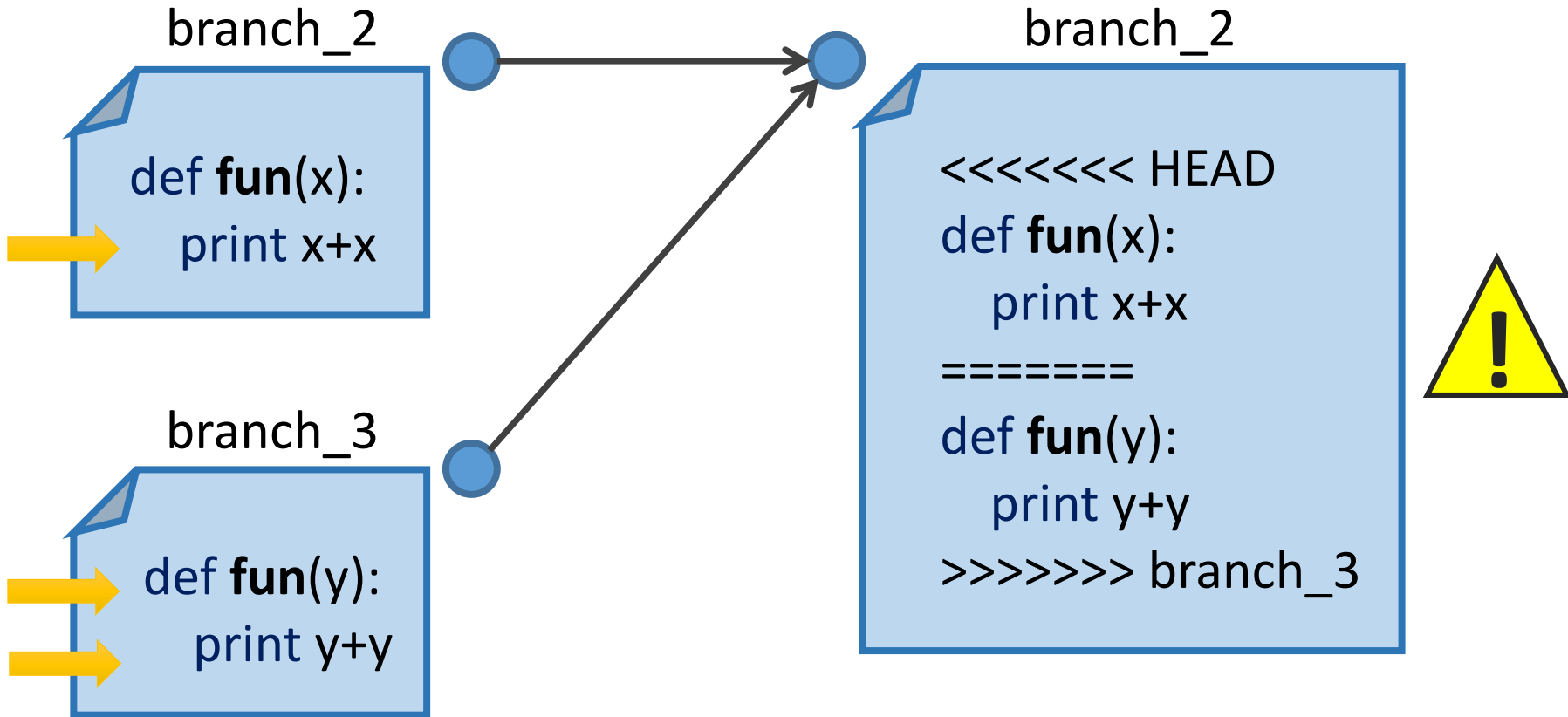
branch_3

Two thick yellow arrows pointing from the left towards the code block.

```
def fun(y):  
    print y+y
```



Mesclando *Commits*





Resolvendo Conflitos

- Alterar o arquivo manualmente
- Utilizar uma interface gráfica
 - kdiff3, tkdiff, meld, xxdiff, vimdiff, [p4merge](#)

Com o p4merge configurado*, basta fazer:

```
$ git mergetool
```

* Veja a seção Configurações



Teste.txt - Perforce P4Merge

File Edit View Search Help

1 diffs (Ignore line ending differences) | Tab spacing: 4 | File format (Encoding: System Line endings: Windows)

Base: Teste.txt.BASE.8400.txt
Left: Teste.txt.LOCAL.8400.txt Differences from base: 0
Right: Teste.txt.REMOTE.8400.txt Differences from base: 0
Merge: Teste.txt Conflicts: 1

./Teste.txt.LOCAL.8400.txt ./Teste.txt.BASE.8400.txt ./Teste.txt.REMOTE.8400.txt

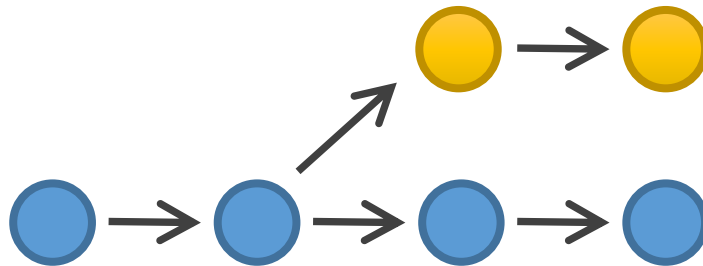
1	def fun(x) :	1	def fun(x) :	1	def fun(y) :
2	print x+x	2	print x	2	print y+y
3		3		3	

Teste.txt

```
def fun(x) :
    print x
def fun(x) :
    print x+x
def fun(y) :
    print y+y
```

```
$ git commit -a
```

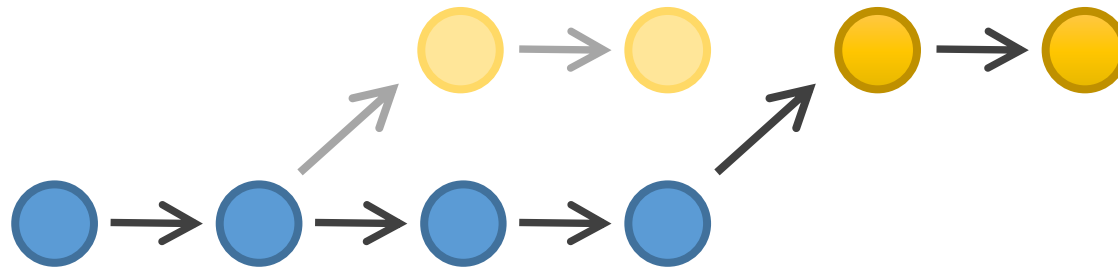
Rebase



```
$ git rebase <base> [-i]
```

Replica os *commits* do *branch* <base> para o atual. Na forma iterativa é possível escolher entre manter, omitir ou editar um *commit*.

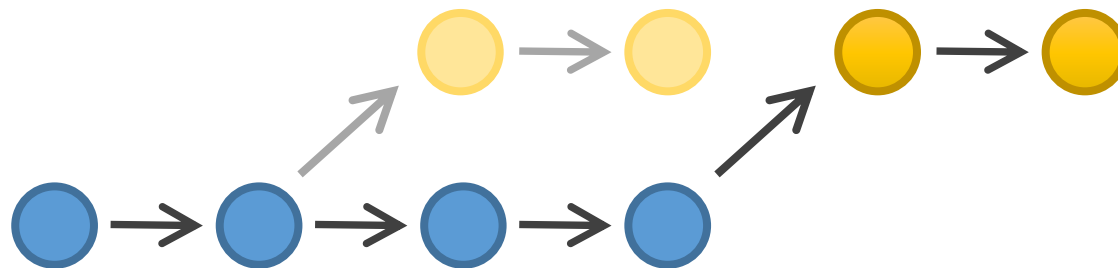
Rebase



```
$ git rebase <base> [-i]
```

Replica os *commits* do *branch* <base> para o atual. Na forma iterativa é possível escolher entre manter, omitir ou editar um *commit*.

Rebase

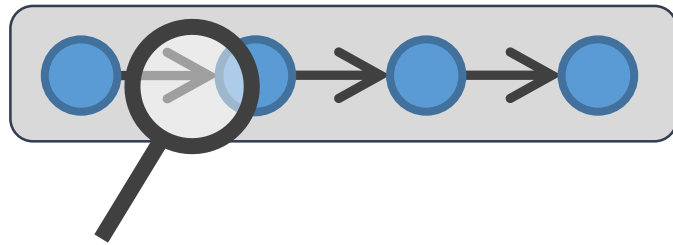


```
$ git rebase <base> [-i]
```

Caso haja algum conflito:

```
$ git mergetool
```

```
$ git rebase --continue
```



Analizando o Log

Analizando o *Log*

```
$ git shortlog
```

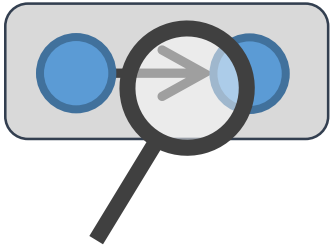
Exibe a primeira linha dos *commits* que cada autor enviou.

```
$ git shortlog -s
```

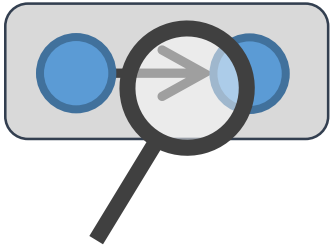
Exibe o número de *commits* que cada autor enviou.

```
$ git shortlog -n
```

Exibe, em ordem numérica, o número de *commits* que cada autor enviou.



Analizando o *Log*



```
$ git log
```

Exibe o *log* de *commits*.

```
$ git log -<n>
```

Exibe os últimos <n> *commits*.

```
$ git log --since==<date>
```

Exibe os *commits* desde a data <date>.

Ex: “3.weeks”, “yesterday”, “3.minutes”

Analizando o *Log*

```
$ git log --graph
```

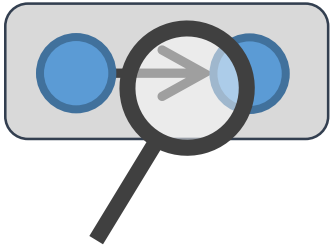
Exibe o *log* em forma de gráfico.

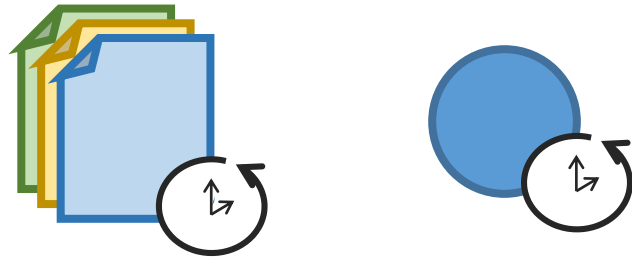
```
$ git log --oneline
```

Exibe o *log*, um *commit* (abreviado) por linha.

```
$ git log --all
```

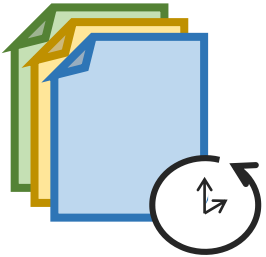
Exibe o *log* de todas as *tags*, *branches*, ...





Desfazendo Ações

Recuperando Arquivos



```
$ git checkout [--] <arquivo>
```

Recupera o arquivo <arquivo> do último *commit*.

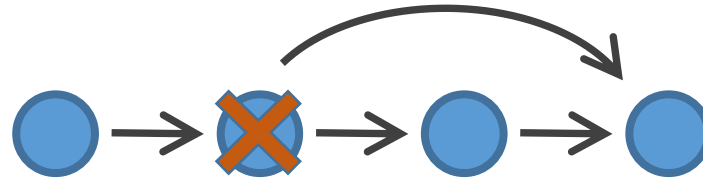
```
$ git checkout <commit> <arq>
```

Recupera o arquivo <arq> do *commit* <commit>.

```
$ git checkout <commit>
```

Recupera os arquivos do *commit* <commit>.

Revertendo *Commits*

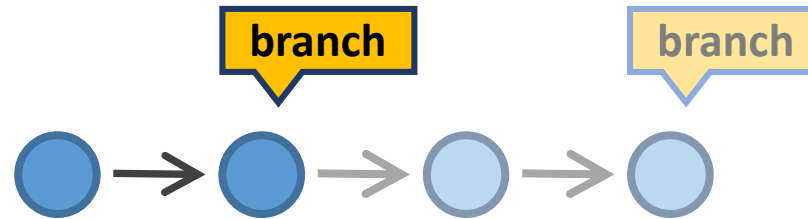


```
$ git revert <commit>
```

Cria um novo *commit* no *branch* atual que desfaz o que foi introduzido no *commit* <commit>.

- Consertar um *bug* introduzido por um *commit*.
- Não remove o *commit* <commit>

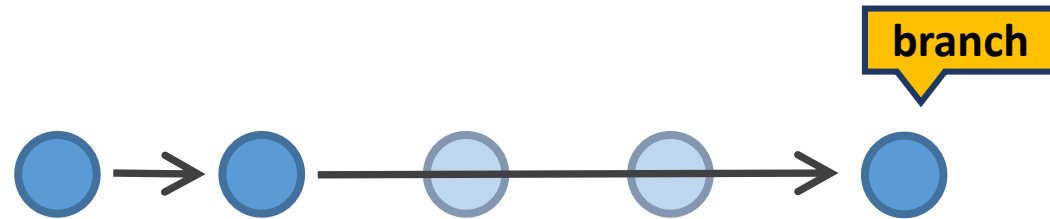
“Excluindo” *Commits*



```
$ git reset --soft <commit>
```

Altera apenas o HEAD para o *commit* <commit>. Não altera a área transitória nem o diretório de trabalho.

“Excluindo” *Commits*

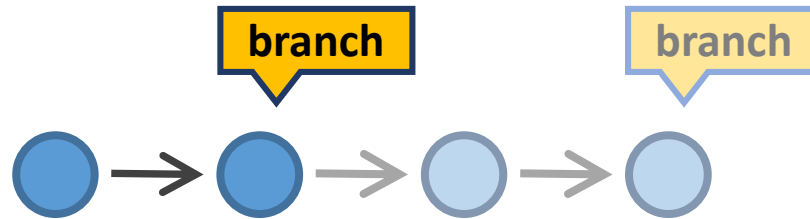


```
$ git reset --soft <commit>
```

```
$ git commit
```

Substitui os *commits* por um único *commit*. O diretório de trabalho não é alterado.

“Excluindo” *Commits*



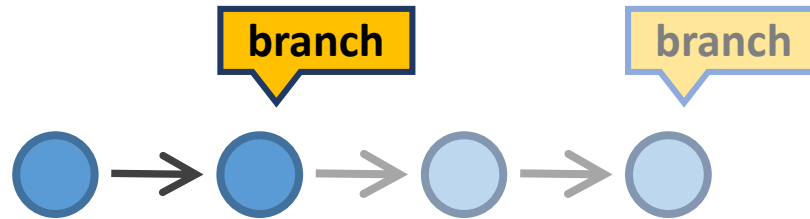
```
$ git reset --hard <commit>
```

Altera a área transitória e o diretório de trabalho para o *commit* <commit>.



O comando *git reset* é uma das poucas formas de se perder informação utilizando o *git*, pois os *commits* deixam de aparecer no *git log*.

“Excluindo” *Commits*



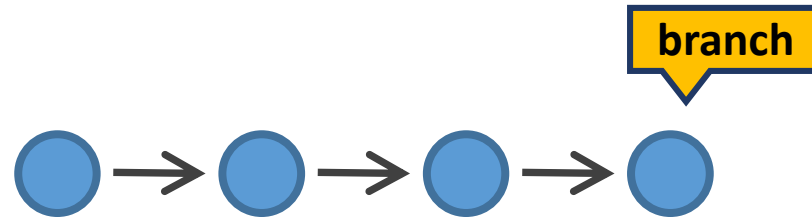
```
$ git reset [--mixed] <commit>
```

Altera apenas a área transitória para o *commit* <commit>. Não altera o diretório de trabalho.

É necessário um *git add* para selecionar os arquivos do diretório que irão para o próximo *commit*, caso contrário irá o arquivo da área transitória.

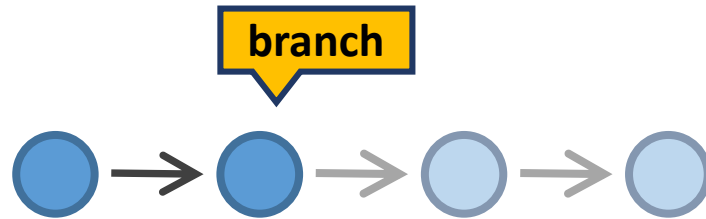


“Excluindo” *Commits*



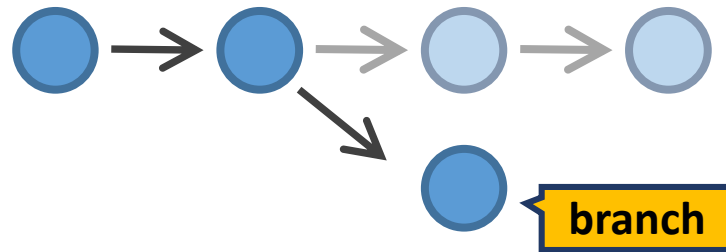


“Excluindo” *Commits*



```
$ git reset <commit>
```

“Excluindo” *Commits*

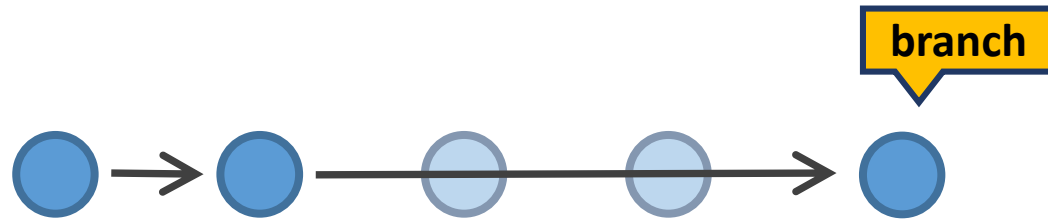


```
$ git reset <commit>
```

```
$ git commit
```

Mantém os arquivos da área transitória, ou seja, do *commit* <commit>.

“Excluindo” *Commits*



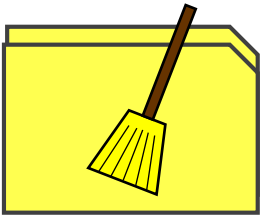
```
$ git reset <commit>
```

```
$ git add <arquivos>
```

```
$ git commit
```

Mantém os arquivos <arquivos> do diretório.

Limpendo o Diretório



```
$ git clean [-f]
```

Exclui os arquivos que não estão sendo rastreados. É possível forçar a exclusão.

```
$ git clean -n
```

Exibe os arquivos não rastreados que serão excluídos.

Configurações Básicas



Configuração Inicial do Git

```
$ git config --global user.name <nome>
```

Atribui <nome> ao nome do usuário.

```
$ git config --global user.email <email>
```

Atribui <email> ao e-mail do usuário.

```
$ git config --global core.editor <editor>
```

Atribui <editor> como editor padrão. Ex.: notepad, emacs ...



Configurando o p4merge

```
$ git config --global merge.tool p4merge
```

Atribui p4merge como ferramenta de mesclagem.

```
$ git config --global mergetool.p4merge.cmd  
"p4merge.exe %BASE% %LOCAL% %REMOTE% %MERGED%"
```

Atribui o caminho e a forma de como executar o programa.

- Analogamente para *diff* e *difftool*.