Controle de Versão

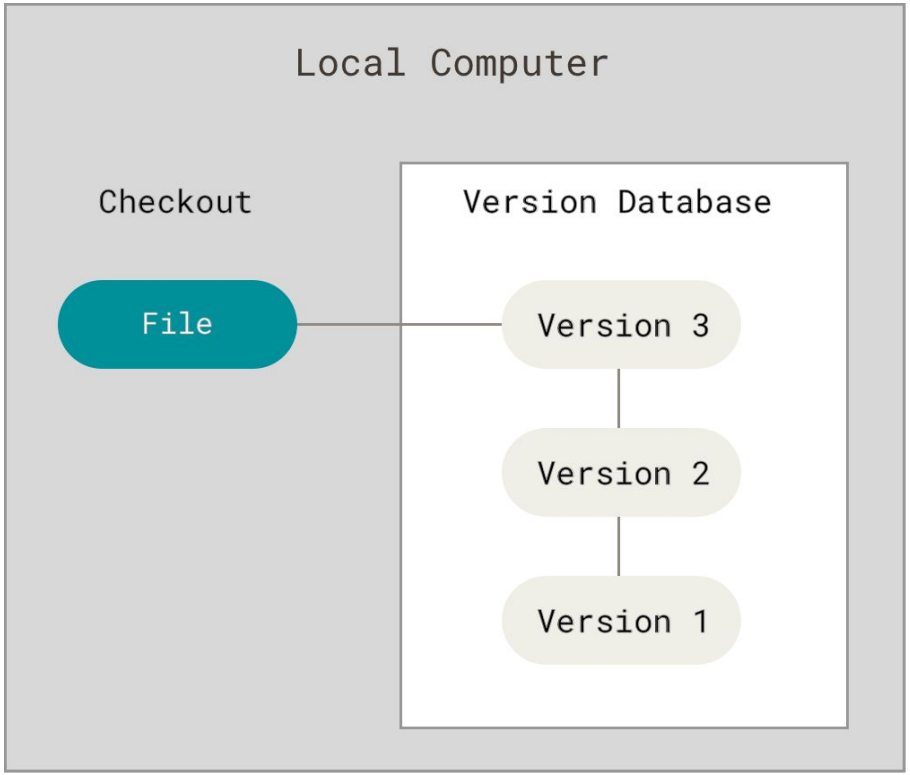
Rafael Lemos
rafaellemos42@gmail.com

- O que é Git?
- O que é GitHub?
- Qual a diferença entre os dois?

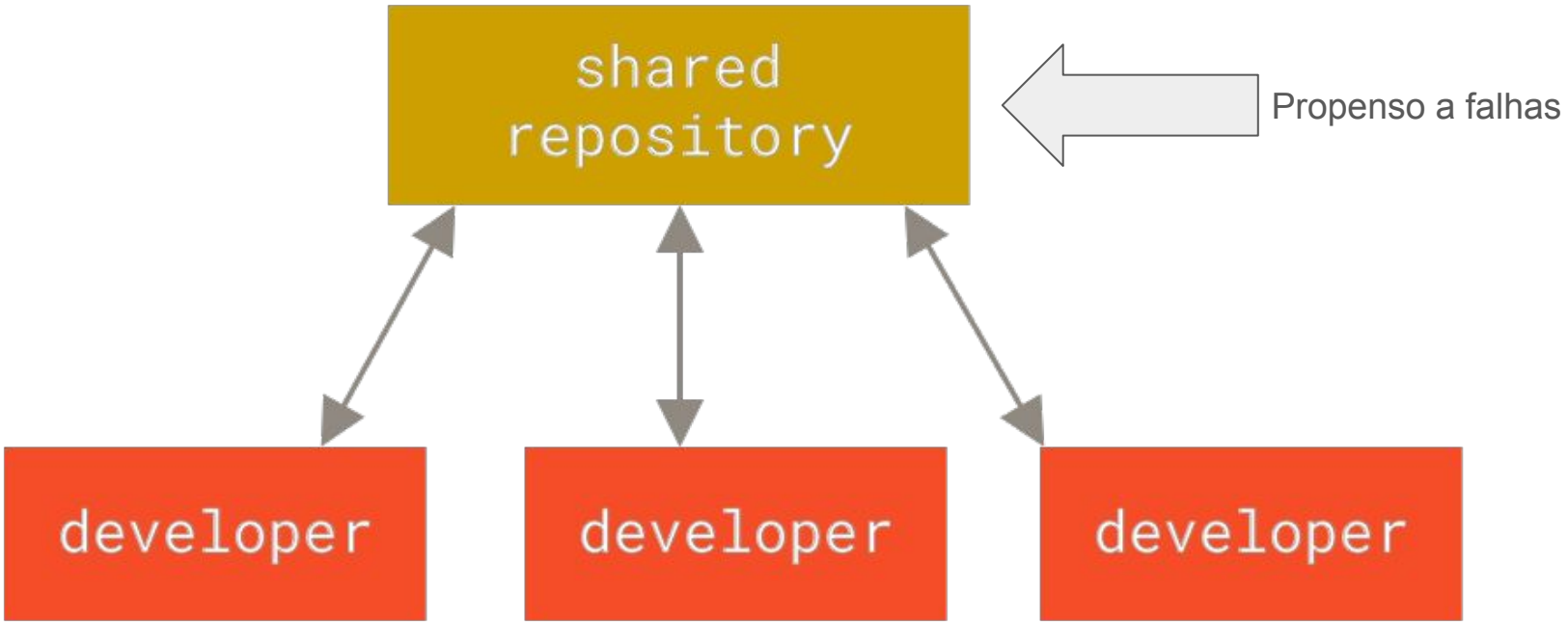
Controle de Versão

Sistema que acompanha e registra todas as alterações feitas em arquivos ou projetos ao longo do tempo. Ele permite reverter arquivos ou projetos a versões anteriores, comparar mudanças feitas em diferentes momentos, e identificar quem fez cada modificação, quando foi feita e por quê. Isso facilita a correção de erros, o trabalho em equipe e o entendimento da evolução do projeto.

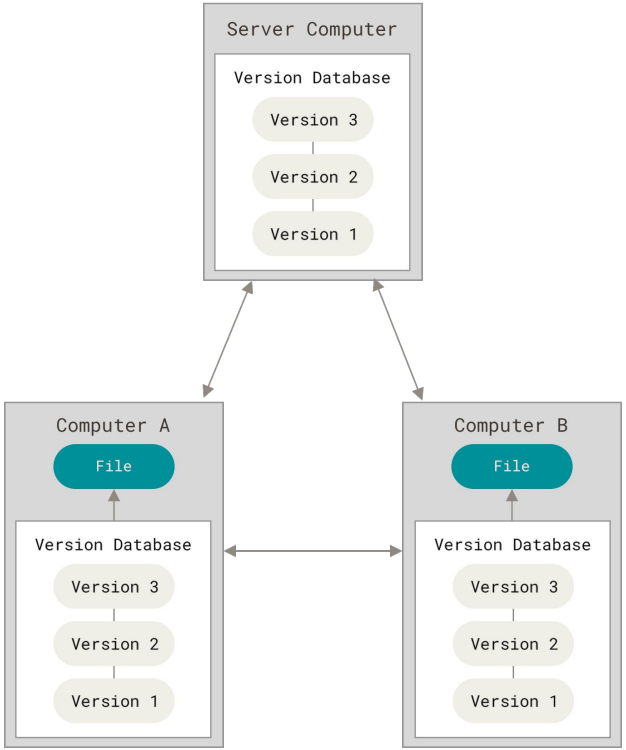
Controle de Versão Local



Controle de Versão Centralizado



Controle de Versão Distribuído



De onde veio o Git?

- Início do Linux (1991-2002): “controle de versão” era o próprio Linus Torvalds e pequenos patches (totalmente contra CVCS);

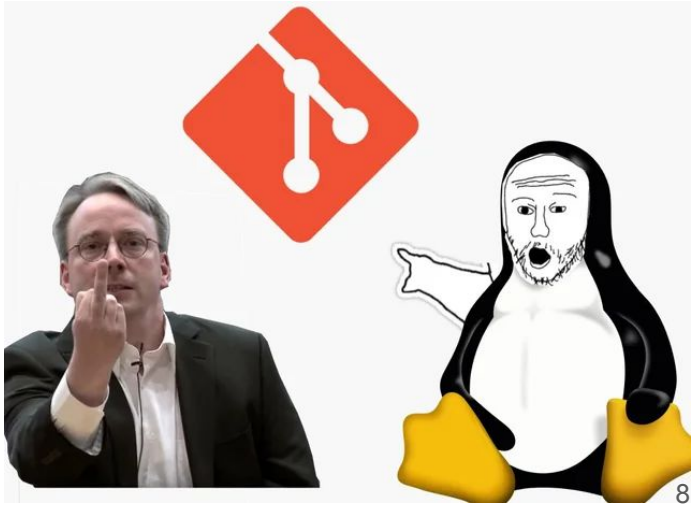
- 2002: incorporação do BitKeeper (DVCS) no Linux.
  - Porém, a licença era extremamente restritiva

BitKeeper

https://graphite.dev/blog/bitkeeper-linux-story-of-git-creation

De onde veio o Git?

- 2005: quebra da licença fazendo engenharia reversa do BitKeeper;
- Quebra da relação entre as comunidades;
- Discussão na comunidade sobre alternativas:
  - Rápidas
  - Design simples
  - Suporte a desenvolvimento não-linear
  - Totalmente distribuídas
  - Capazes de lidar com grandes projetos (Linux)



https://graphite.dev/blog/bitkeeper-linux-story-of-git-creation

De onde veio o Git?

- Linus de “peacemaker”:

```
On Fri, 8 Apr 2005, Andrea Arcangeli wrote:
>
> Why not to use sql as backend instead of the tree of directories?

Because it sucks?

I can come up with millions of ways to slow things down on my own. Please
come up with ways to speed things up instead.

                Linus
```

- Linus entra de férias…

https://graphite.dev/blog/bitkeeper-linux-story-of-git-creation

De onde veio o Git?

- Durante as “férias”:
  - Troca de emails sobre alternativas (criar uma nova era fora de cogitação)
  - Senso de urgência para continuar o desenvolvimento do Linux
- 7 dias depois… Nasce o Git (ou o esqueleto dele)
  - Não havia discussão do Workflow, mas sim da arquitetura (hash-based)
  - Ainda não era 100% um controle de versão

> "git" is really trivial, written in four days. Most of that was not actually spent coding, but thinking about the data structures.
>
> Linus

https://graphite.dev/blog/bitkeeper-linux-story-of-git-creation

De onde veio o Git?

Duas semanas depois:

List:	git
Subject:	First ever real kernel git merge!
From:	Linus Torvalds <torvalds () osdl ! org>
Date:	2005-04-17 22:10:25

Dois anos depois:

De onde veio o Git?

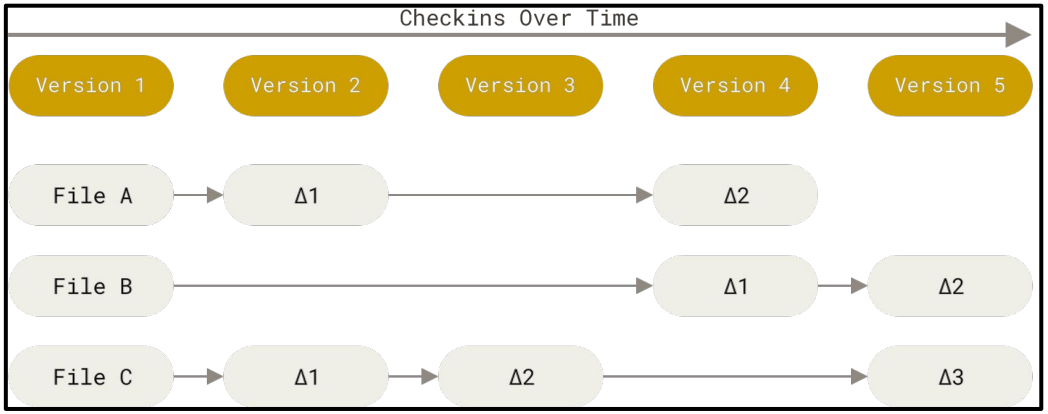
“Git foi criado como uma ferramenta para destravar futuros lançamentos do kernel do Linux, não com a intenção de reinventar globalmente toda a gestão de código-fonte;

Quando pensamos na história, muitas vezes a romantizamos como algo que nasce de um súbito lampejo de inspiração. Mas a criação do Git mostra a realidade muito mais dura da invenção: um desacordo crescente sobre uma licença; a necessidade de uma solução improvisada para destravar o trabalho; e depois, anos e anos de aperfeiçoamento contínuo, liderado não pelo inventor, mas por uma comunidade.”

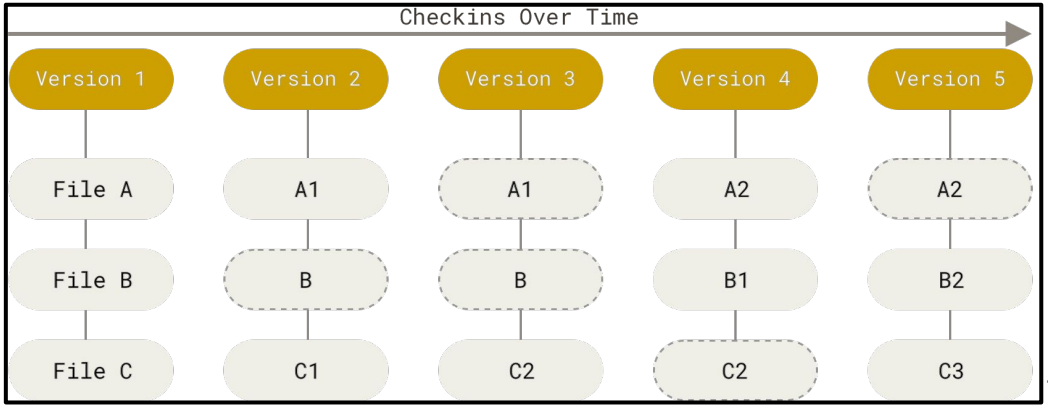
https://graphite.dev/blog/bitkeeper-linux-story-of-git-creation

Tá, mas por que o Git é especial?

Normalmente: *delta-based*



Git: *“snapshot”-based*

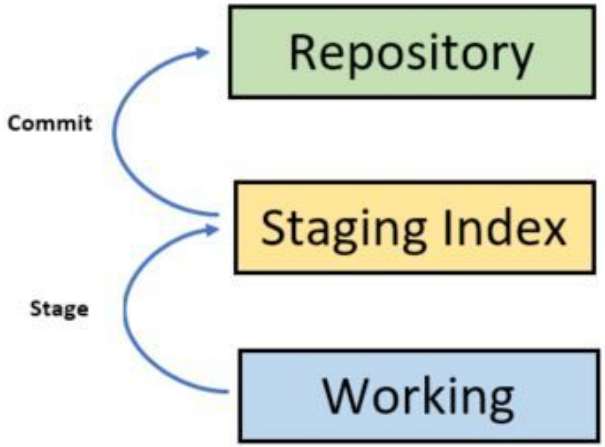


Vantagens

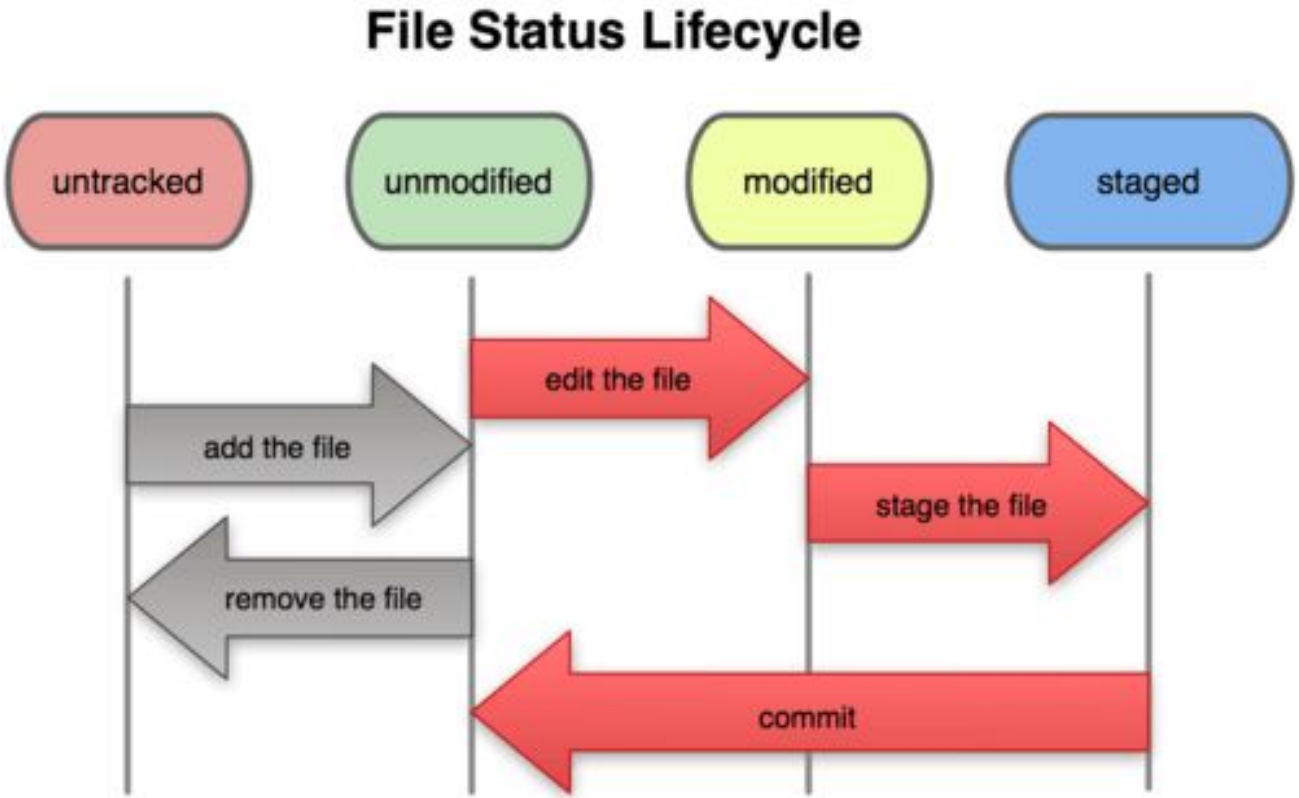
- Distribuído: Cada desenvolvedor tem uma cópia completa do repositório, incluindo todo o histórico.
- Seguro: Usa SHA-1 para garantir integridade dos dados.
- Flexível: Suporta vários fluxos de trabalho (centralizado, com ramificações, etc.).
- Rápido e local: Nenhuma informação é necessária de outro computador da rede.
    - Mudanças e histórico são armazenados localmente nos arquivos (lembram do *hash*?)

Os Três Estados

- *Modified*: arquivo alterado, mas ainda sem *commit*;
- *Staged*: arquivo marcado para fazer parte do próximo *commit*;
- *Committed*: arquivo está armazenado no banco de dados local.

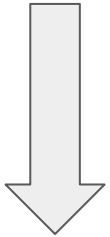


Os Três Estados



Setup Inicial - Git Config

- Sistema (--system): *[path]/etc/gitconfig*
- Global (--global): *~/.gitconfig*
- Repositório (--local): *.git/config*



$ git config --list --show-origin

```
file:/home/rlemos/.gitconfig    user.email=rafael_plemos@hotmail.com
file:/home/rlemos/.gitconfig    user.name=rplemos
file:/home/rlemos/.gitconfig    init.defaultbranch=main
file:/home/rlemos/.gitconfig    credential.helper=store
file:.git/config        core.repositoryformatversion=0
file:.git/config        core.filemode=false
file:.git/config        core.bare=false
file:.git/config        core.logallrefupdates=true
```

Setup Inicial - Git Config

- *$ git config --global user.name “usuário”*
- *$ git config --global user.email “email”*
- *$ git config --global init.defaultBranch main*

- *$ git config --list*

Setup Inicial - Ajuda

- **Geral:** *$ git help*
- **Detalhada:** *$ git help <comando>*
- **Resumida:** *$ git <comando> -h*

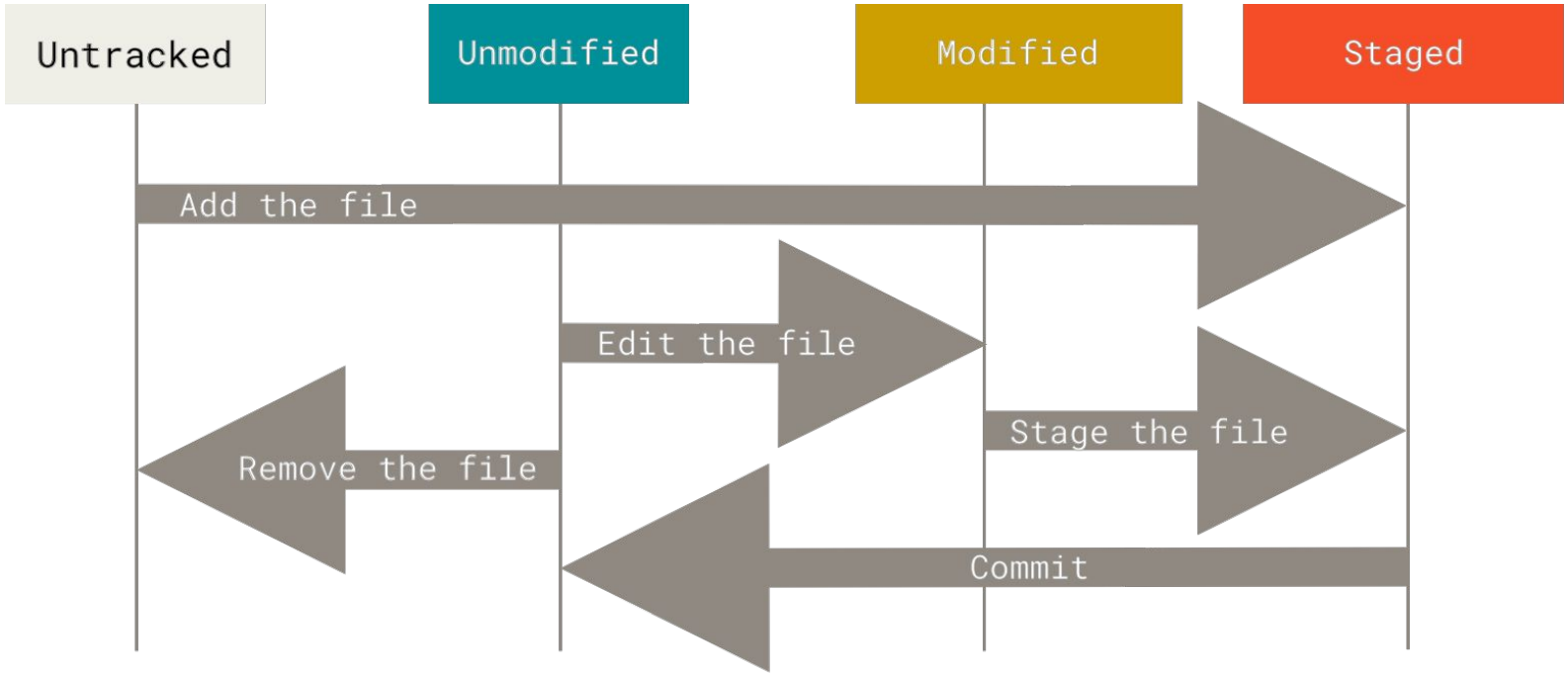
Criando Repositórios Locais

- Criando repositório do zero:
  - *$ git init <pasta>*
- Adicionando arquivos (*tracking*):
  - *$ git add <arquivo>*
- *“Committando”* arquivos (só os que foram adicionados):
  - *$ git commit -m <mensagem>*
- Checando status:
  - *$ git status*

Status dos Arquivos

- ***Tracked***:
  - Arquivos que o Git “sabe que existem”;
  - Podem ser ***unmodified***, ***modified*** e ***staged***;
- ***Untracked***:
  - Arquivos que o Git “não sabe que existem”;
  - Precisam ser adicionados;
- ***Ignored:***
  - Arquivos de log, configuração…
  - Presentes no arquivo *.gitignore*
  - Boas práticas e templates: *https://github.com/github/gitignore*

Status dos Arquivos



Histórico de Commits

$ git log

```
commit af4750e9cbe1030dfdd29637b6a32da2b63de13e (HEAD -> main, origin/main, origin/HEAD)
Author: rplemos <rafael_plemos@hotmail.com>
Date:   Fri Jul 4 18:43:02 2025 -0300

    Small fixes

commit eaf75a91c416c2eb41d390002848859f6ccbee6a
Author: rplemos <rafael_plemos@hotmail.com>
Date:   Thu Jul 3 07:48:37 2025 -0300

    Fix redundancy for electrostatic contacts

commit 7c33868feb0cb152fed2ce541b955737a04c5a18
Author: rplemos <rafael_plemos@hotmail.com>
Date:   Mon Jun 30 12:57:27 2025 -0300

    Added -c flag to select specific chains

commit 0cafb87fc162ee8d9e64a41868007c6642289cdf
Author: rplemos <rafael_plemos@hotmail.com>
Date:   Tue Jun 24 20:47:46 2025 -0300

    Fix edge case in .cif NMR pH values

commit 8d4fe2c589422de35da4aaafdbab246082c8e636
Author: rplemos <rafael_plemos@hotmail.com>
Date:   Mon Jun 23 16:27:34 2025 -0300

    Fixed output format for web
:...skipping...
```

Histórico de Commits

$ git log --stat

```
commit af4750e9cbe1030dfdd29637b6a32da2b63de13e (HEAD -> main, origin/main, origin/HEAD)
Author: rplemos <rafael_plemos@hotmail.com>
Date:   Fri Jul 4 18:43:02 2025 -0300

    Small fixes

 src/contacts.py | 18 +++++++++---------
 src/parser.py   |  2 +-
 2 files changed, 10 insertions(+), 10 deletions(-)

commit eaf75a91c416c2eb41d390002848859f6ccbee6a
Author: rplemos <rafael_plemos@hotmail.com>
Date:   Thu Jul 3 07:48:37 2025 -0300

    Fix redundancy for electrostatic contacts

 src/contacts.py | 23 ++++++++++++++++++++++-
 1 file changed, 22 insertions(+), 1 deletion(-)

commit 7c33868feb0cb152fed2ce541b955737a04c5a18
Author: rplemos <rafael_plemos@hotmail.com>
Date:   Mon Jun 30 12:57:27 2025 -0300

    Added -c flag to select specific chains

 cocada.py        | 14 ++++++++++----
 src/argparser.py | 22 ++++++++++++++++++++--
 src/classes.py   |  3 ++-
 src/contacts.py  |  5 ++++-
 src/process.py   |  2 +-
 5 files changed, 38 insertions(+), 8 deletions(-)
```

Histórico de Commits

$ git log --oneline

```
af4750e (HEAD -> main, origin/main, origin/HEAD) Small fixes
eaf75a9 Fix redundancy for electrostatic contacts
7c33868 Added -c flag to select specific chains
0cafb87 Fix edge case in .cif NMR pH values
8d4fe2c Fixed output format for web
a0f1999 Added -d parsing for web, moved logic from main to argparser
cb20697 Minor fixes
b8d6ff9 Full code refactor, ph and silent flags, etc.
34ffdfd Minor changes
d8b7c10 Fixed AS atomnames in proteins with hydrogens
b95121e For web, changing custom_distances to be passed by a flag instead of from a
4ca89a1 Merging CL development branch into this repository. CL and Web version now
beba1d8 Aromatics distances now can be user-defined
9108b4e Fixed a few rare bugs
7378ff2 Fixed small bugs with entities
0d2e941 First iteration of flexible distances defined by the user
317077b (old-origin/main, old-origin/HEAD) Preliminary interface contact detection
4cf0a92 Main refactoring to include parameters in a new Context class
858f9e6 Modified version for the Web application
9609b9b (tag: v1.0) Added citation
c8c6918 Fixed a few bugs with parser and output
9c8c961 Minor changes and web especific functions
c1afa7f Fixed edge cases on title and occupancy, and a few other small bugs
```

Diff

- Mostra as diferenças entre versões de arquivos no repositório;
- Útil para ver o que mudou entre commits.

```
diff --git a/src/classes.py b/src/classes.py
index 70de43d..4f5d464 100644
--- a/src/classes.py
+++ b/src/classes.py
@@ -227,7 +227,10 @@ class Contact:
             "disulfide_bond":"DS",
             "stacking-other":"AS",
             "stacking-parallel":"AS", # on v.1 all aromatic stackings will be considered the same
-            "stacking-perpendicular":"AS" # need to reimplement later
+            "stacking-perpendicular":"AS", # need to reimplement later
+            "uncertain_attractive": "uAT",
+            "uncertain_repulsive": "uRE",
+            "uncertain_salt_bridge": "uSB"
         }

         all_values = list(self.__dict__.values())
diff --git a/src/contacts.py b/src/contacts.py
index 8de75ba..266534d 100644
--- a/src/contacts.py
+++ b/src/contacts.py
```

Repositórios Remotos

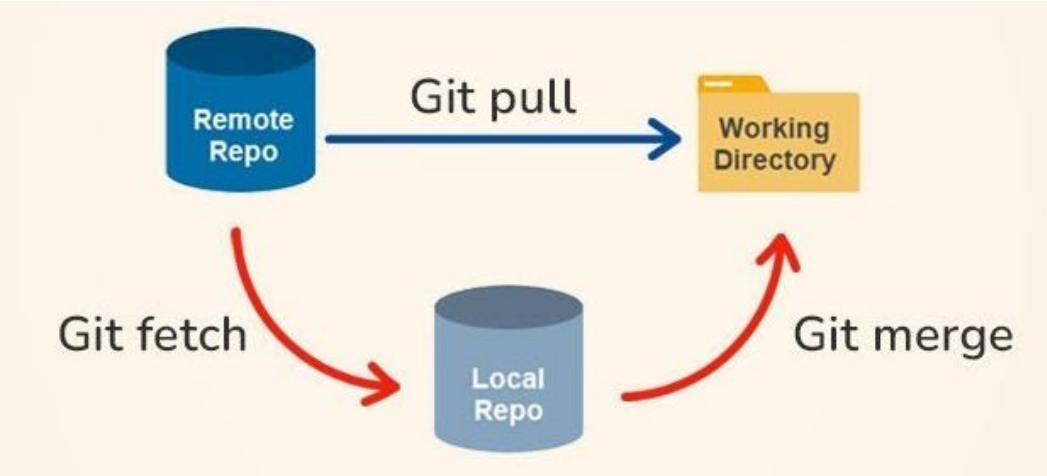
- Integração especialmente com GitHub;
- Opções:
  - Criar repositório local e adicioná-lo a um remoto;
  - Clonar repositório já existente.

Clonando Repositórios

- Clonando repositório (com todas as informações):
  - *$ git clone <repositório> <pasta_local>*
- Checando informações:
  - *$ git remote -v*
- Adicionando outros repositórios remotos
  - *$ git remote add <nome> <url>*
- Checando status:
  - *$ git status*

Fetch, Pull e Merge

- *$ git fetch:* busca atualizações no repositório remoto;
- *$ git merge:* une as atualizações do repositório remoto com o local;
- *$ git pull:* fetch + merge.



Exercício

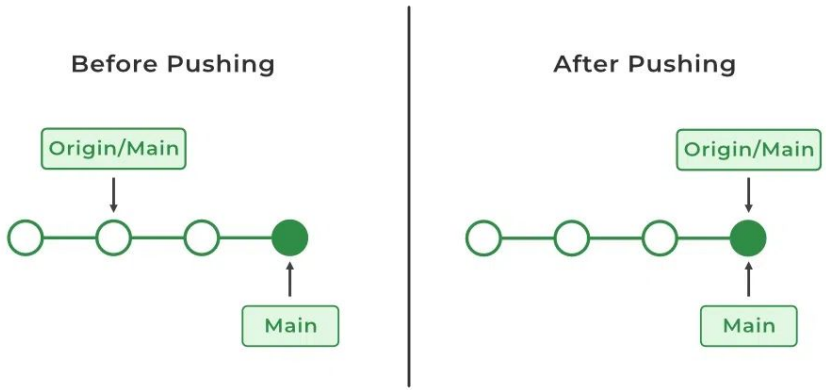
- Criar repositório no GitHub
    - Adicionar .gitignore, README e Licença
- Adicionar como ‘origin’ no repositório local
- Unir os dois repositórios localmente
- Como resolver os conflitos?

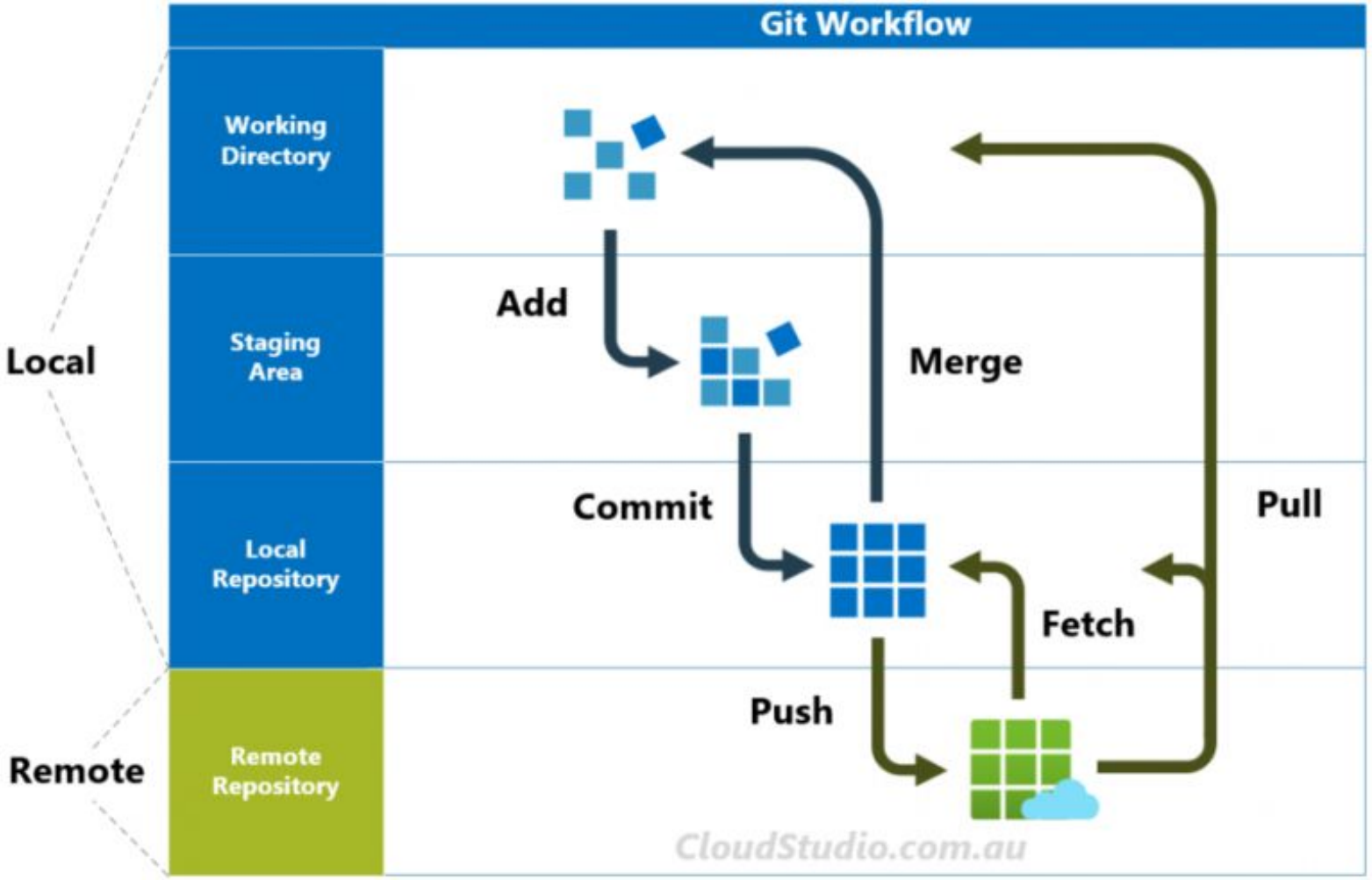
Exercício

- Criar repositório no GitHub
  - Adicionar .gitignore, README e Licença
- Adicionar como ‘origin’ no repositório local
- Unir os dois repositórios localmente
- Como resolver os conflitos?
  - *$ git merge origin/main --allow-unrelated-histories* (força um merge)
  - *$ git reset --hard origin/main* (sobrescreve o local)
  - *$ git clone*

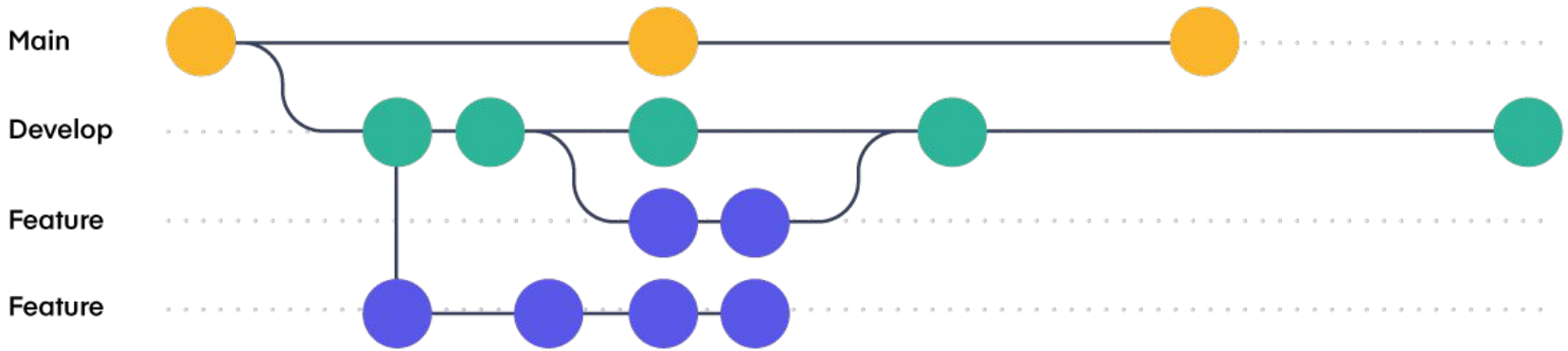
Push

- *$ git push <remote> <branch>*
- Envia commits locais (que já estão salvos no seu repositório local) para um repositório remoto.




Git Workflow
Working Directory
Staging Area
Local Repository
Remote Repository
Local
Remote
Add
Commit
Push
Merge
Fetch
Pull
CloudStudio.com.au

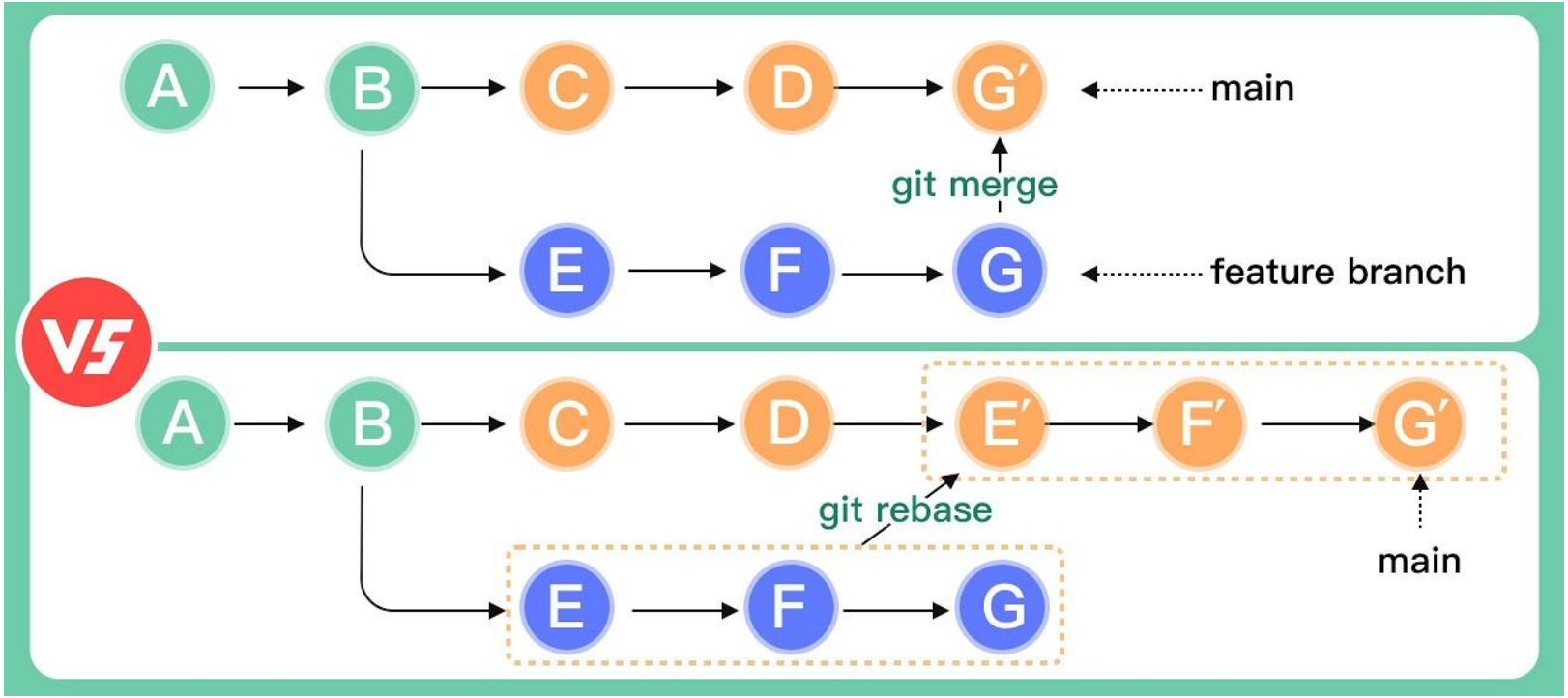
Branches



Branches

- Checar branches:
  - *$ git branch* (mostra locais)
  - *$ git branch -r* (mostra remotos)
  - *$ git branch -a* (mostra todos)
- Criar novo branch:
  - *$ git branch <branch_name>*
- Trocar branch:
  - *$ git switch <branch_name>*
- Juntar branches:
  - *$ git merge <outra_branch> <branch_atual>*

Merge vs. Rebase

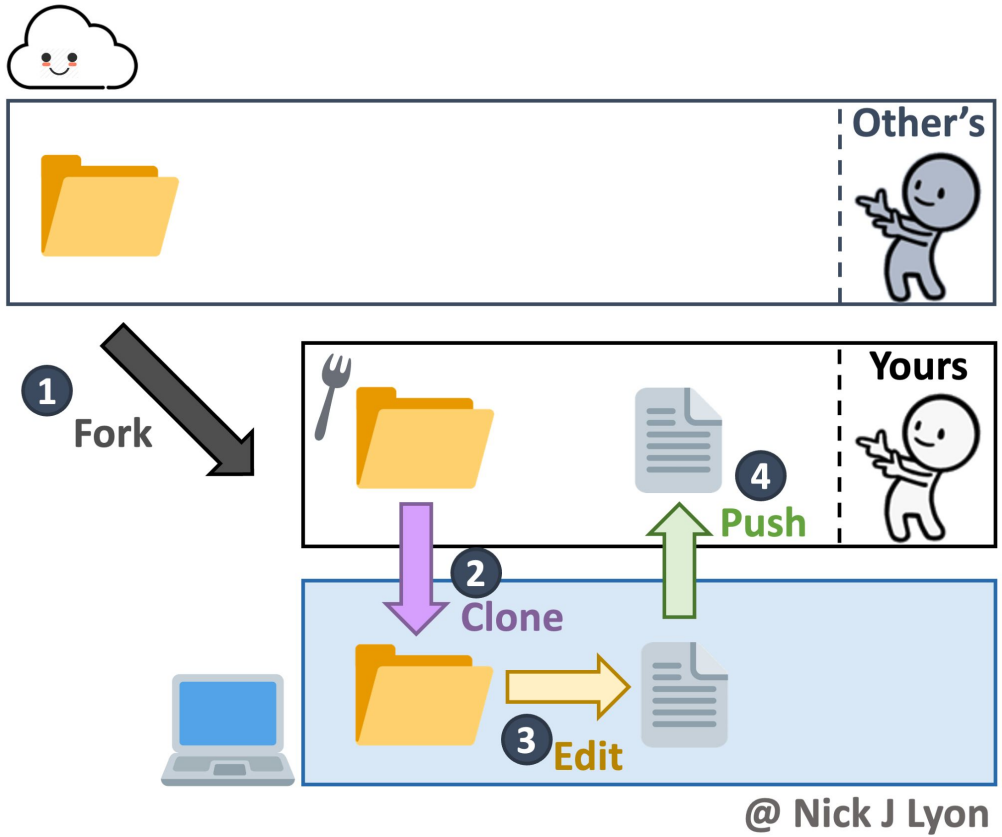


Rebase

- Histórico linear: Mais fácil de entender e revisar.
- Evita commits de merge desnecessários.
- Ótimo para preparar código antes de subir para o repositório remoto.

- **Nunca** use rebase em branches já publicados/remotos que outras pessoas estão usando — isso muda a história e pode causar conflitos para os outros.

Forks



Forks

- Cópia independente de um repositório;
- Copia todo o histórico de commits, branches, arquivos e estrutura do projeto original;
- Serve para experimentar, modificar ou contribuir sem afetar diretamente o projeto principal (o "upstream").

Forks

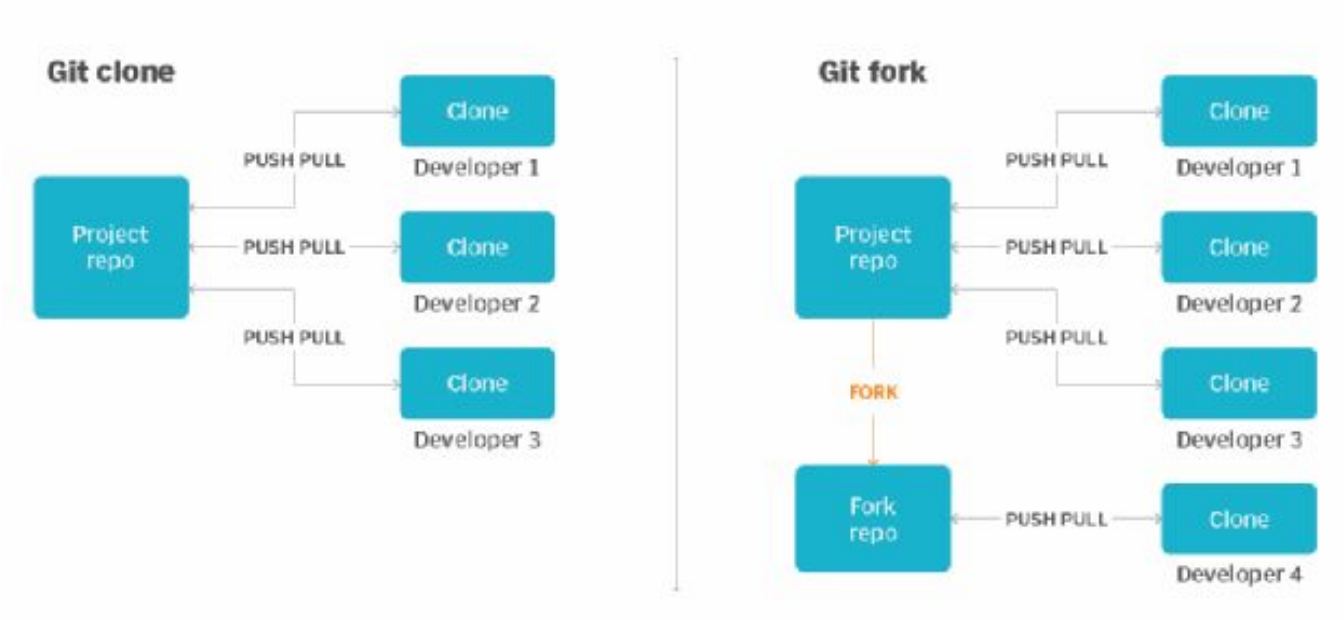
🧠 Quando você usaria um fork?

Situação	Por quê usar fork
Contribuir com projetos open-source	Você não tem permissão de escrita no repositório original
Customizar um projeto existente	Você quer fazer mudanças próprias sem depender do autor original
Testar ideias sem riscos	Pode alterar livremente, criar branches, apagar, reescrever histórico

Clones vs. Forks

Clone: Cópia local. Baixa os arquivos de um repositório remoto.

Fork: Cópia na nuvem. Cria um novo repositório online.

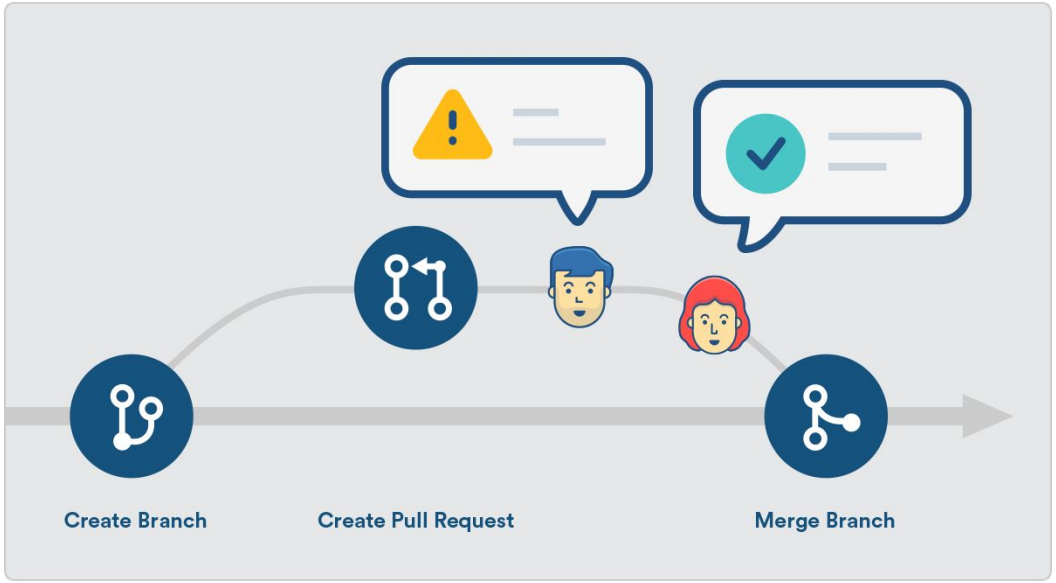


Clones vs. Forks

- Fork no GitHub;
- Clone do Fork localmente;
  - Adicionar o original como remote (*$ git remote add upstream <.git>*)
- Criar Branch de trabalho;
- Commit das alterações;
- Push para o Fork no GitHub;
- Pull Request para o repositório original.

Pull Requests

- Pedido formal para que alterações feitas em um branch (ou um fork) sejam mescladas a outro branch.



Pull Requests

- Alterações em uma branch separada (ou um fork);
- Envio para o repositório remoto (branch ou fork);
- Abertura de um pull request comparando sua branch com a branch de destino.
- Outros colaboradores podem:
  - Ver o que foi alterado (diffs),
  - Fazer comentários linha a linha,
  - Aprovar ou solicitar mudanças,
  - E, por fim, dar merge no PR.

Pull Requests

Quando usar?

- Colaborando com projetos open-source.
- Requisitando revisão de código.
- Gerenciando alterações entre equipes.
- Trabalhando com forks (sem acesso direto ao repositório principal).

Selecting and operating an ACME client yourself

If your hosting provider does not handle getting and managing certificates for you, and if you have the ability to run commands on your server with sufficient privileges, you can select an ACME client and run it yourself to get certificates from Let's Encrypt.

For most people we recommend the Certbot ACME client. The Certbot website has excellent documentation and instructions for operating Certbot.

There are many more options for ACME client software if for some reason Certbot does not meet your needs.

If your client needs to be configured with the Let's Encrypt ACME API endpoint, it is:

```
https://acme-v02.api.letsencrypt.org/directory
```

We recommend running tests against our staging API first.