
git

...


GitLab


GitHub


Bitbucket

git

News › After controversy, Torvalds begins work on "git"

After controversy, Torvalds begins work on "git"

Following controversy, Linux creator Linus Torvalds has begun writing new software, called "git," to manage the Linux kernel development process.

Robert McMillan (IDG News Service) on 20 April, 2005 09:56

- criado em 2005 por Linus Torvalds
- para manter o código fonte do Kernel do Linux
- substituiu o BitKeeper
- sistema **distribuído** de controle de versões de software

Objetivos

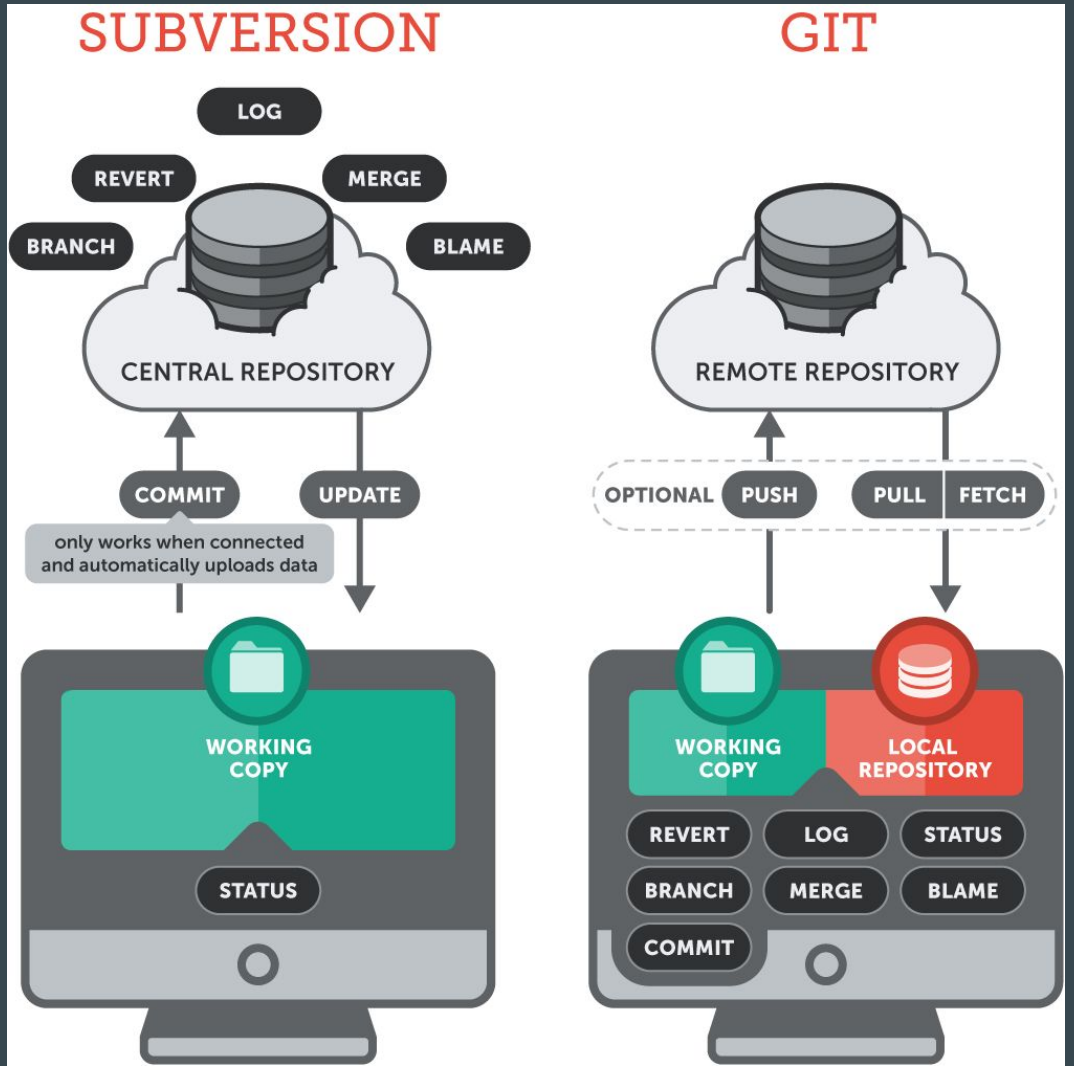
- velocidade
- design simples
- suporte robusto a desenvolvimento não linear (múltiplos branches paralelos)
- **distribuído / offline**
- grandes projetos

Limpando a mente

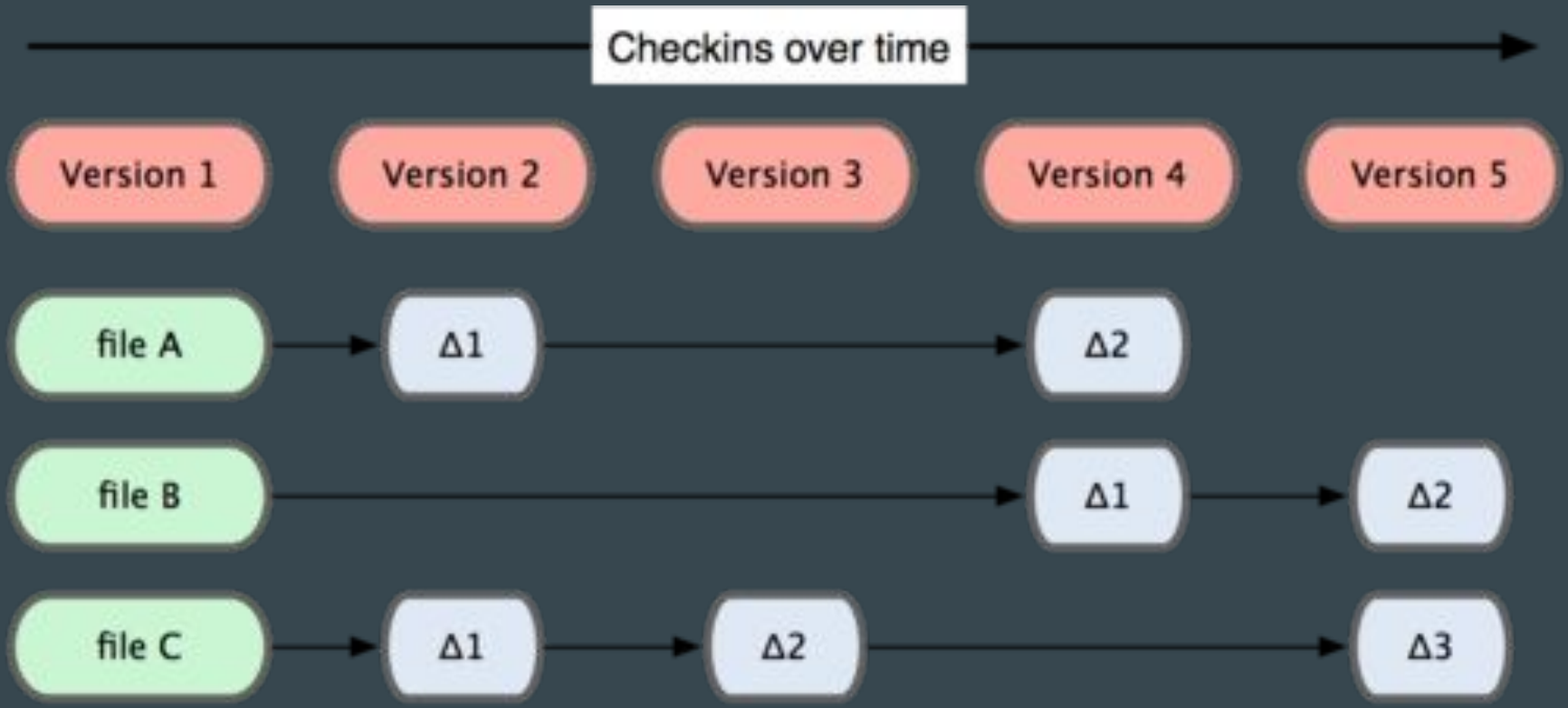
- snapshots
- integridade de revisões (SHA-1)
- controle de revisões local
- operações locais (commit, tag, branch, diff, log, checkout)
- operações remotas (clone, push, pull, fetch)

O SHA-1 gerado é uma string de 40 caracteres hexadecimal, baseado no conteúdo dos arquivos e pastas.

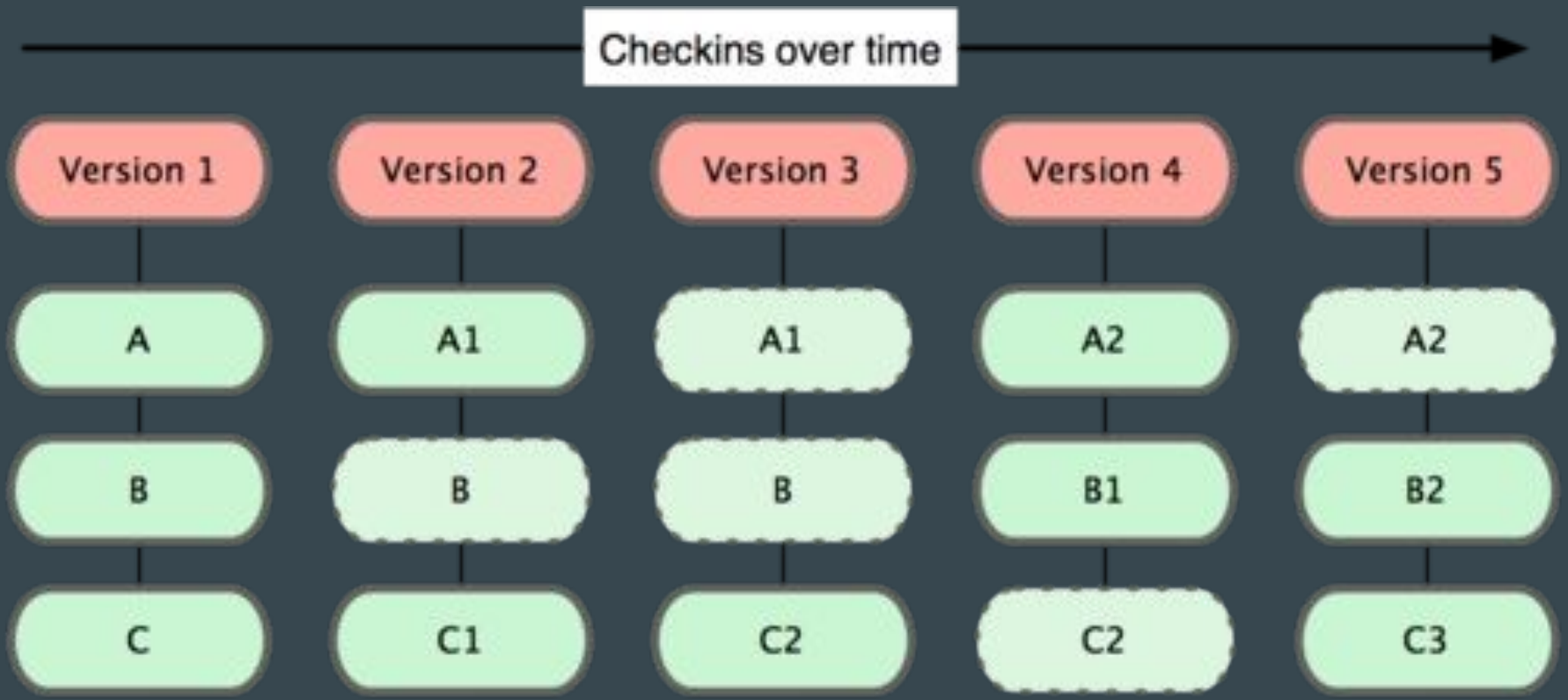
subversion != git



svn - delta-based version control



git - snapshots



git - Instalação

https://git-scm.com/downloads

https://mirrors.edge.kernel.org/pub/software/scm/git/

Ferramentas adicionais sugeridas:

- Meld: http://meldmerge.org/
- Visual Studio Code: https://code.visualstudio.com/
- Git extensions: https://github.com/gitextensions/gitextensions
- TortoiseGit: https://tortoisegit.org/

Configuração do ambiente do usuário

```
Configurações do projeto (git config --local)
Configurações do usuário (git config --global)
Configurações do sistema (git config --system)
```

Configuração do ambiente do usuário

```
git config --global user.name "Fulano de Tal"
git config --global user.email "fulano.de.tal@dominio.com"
git config --global core.editor "vim"
git config --global diff.tool vimdiff
git config --global merge.tool meld
git config --global mergetool.meld.path=”C:\Program Files
(x86)\Meld\Meld.exe"
git config --list
git config --system list
```

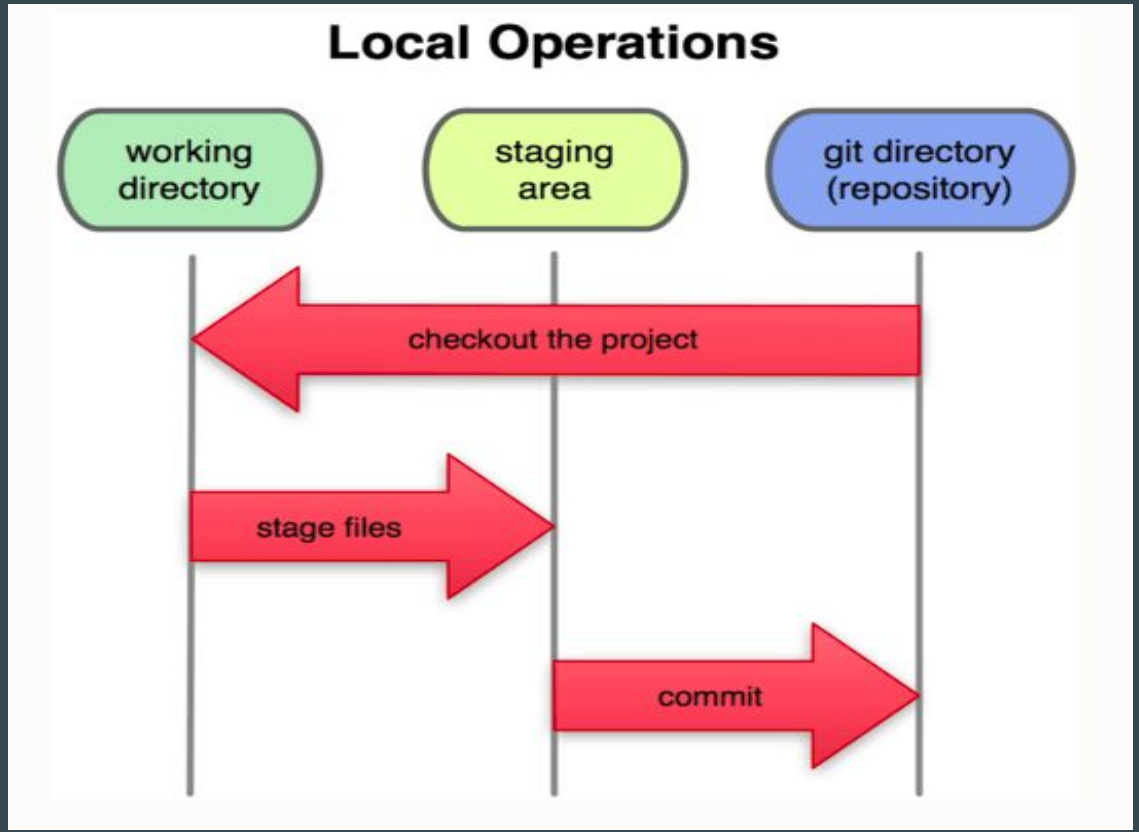
https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup

Iniciando um repositório local

```
mkdir projeto
cd projeto
git init
```

```
git init projeto
cd projeto
```

Conceitos - operações locais



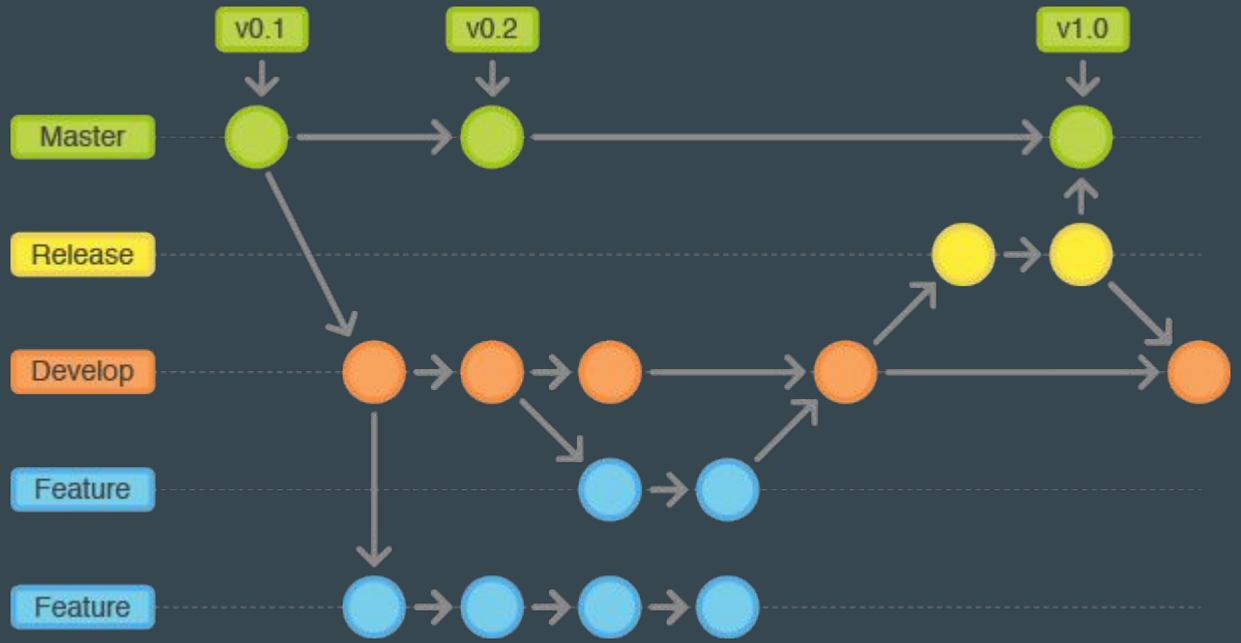
Adicionando arquivos e fazendo o primeiro commit

criar o arquivo README.md

```
git status

git add README.md

git commit -m 'adicionando primeiro arquivo'

git status

git log
```

Conceitos

- branch
- commit
- tag v0.x
  - simples
  - anotada



Branches

git branch <novo-branch>

git branch <novo-branch> <branch-existente>

git branch

git branch -a

git branch -v

git branch -d <branch-remover>

Branches - checkout

git checkout atualiza a área de trabalho com os arquivos de determinado branch ou revisão:

git checkout -b desenvolvimento master

git checkout master

git checkout desenvolvimento

git checkout -b bugx v1.2.3

Branches - merge

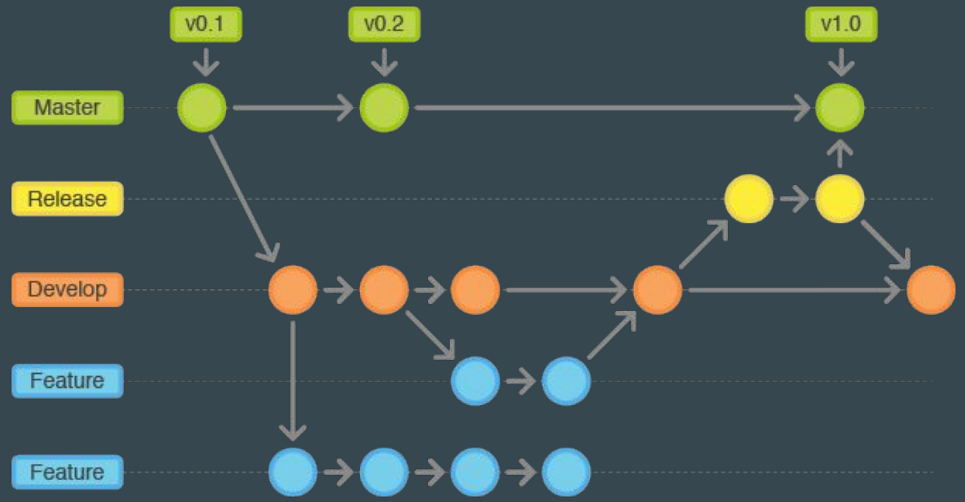
git merge <commit>

git merge master

git merge --squash <branch>

git merge desenvolvimento

git mergetool --tool=meld



Branches - merge conflict

```
$ git checkout meu-branch
Switched to branch 'meu-branch'

$ echo 99 > ARQ1.txt
$ git commit -am 'Adicionando 99 ao arquivo'
[meu-branch 95a8182] Adicionando 99 ao arquivo
 1 file changed, 1 insertion(+), 12 deletions(-)

$ git checkout master
Switched to branch 'master'

$ echo 100 > ARQ1.txt
$ git commit -am 'Arquivo contém 100'
[master 14c0518] Arquivo contém 100
 1 file changed, 1 insertion(+), 11 deletions(-)
```

Branches - merge conflict

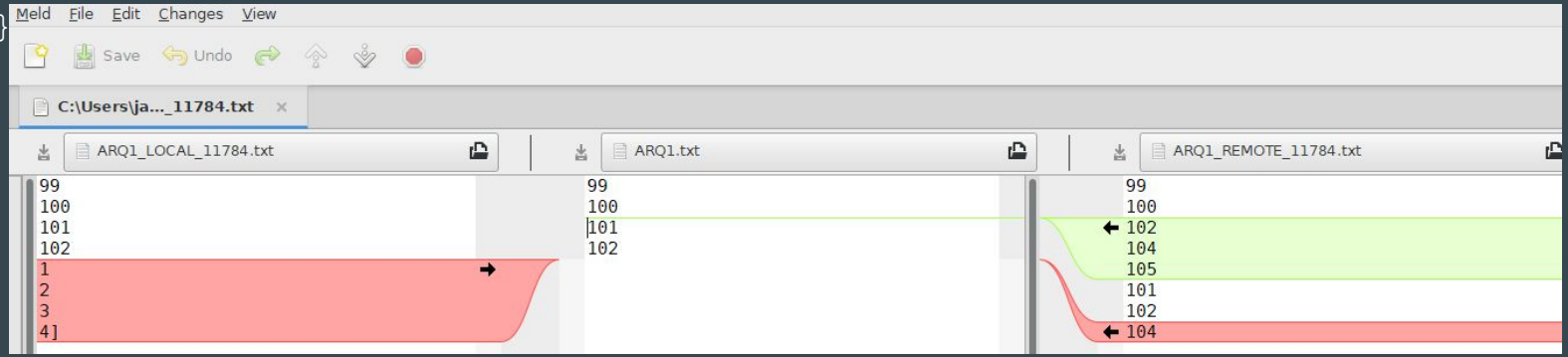
```
$ git merge meu-branch
Auto-merging ARQ1.txt
CONFLICT (content): Merge conflict in ARQ1.txt
Automatic merge failed; fix conflicts and then commit the result.

$ cat ARQ1.txt
<<<<<<< HEAD
100
=======
99
>>>>>>> meu-branch

$ vi ARQ1.txt
```

Branches - merge conflict

```
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   ARQ1.txt

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

$ git commit -am "Merge feito - conflito resolvido"
```

Branches - merge conflict - meld

```
$ git config merge.tool meld
$ git config --global mergetool.meld.path "C:\Program Files
(x86)\Meld\Meld.exe"
$ git mergetool
Merging:
ARQ1.txt

Normal merge conflict for 'ARQ1.txt':
  {local}: modified file
  {remote}
```



Utilizando branches (1)

```
git init teste && cd teste
echo 1 > README1.txt
git add README1.txt
git commit -m 'add README1.txt'

$ git log --oneline --graph --all --decorate
* ed9bf44 (HEAD -> master) add README1.txt
```

Utilizando branches (2)

```
git checkout -b funcionalidade
echo 2 > README2.txt
git add README2.txt
git commit -m 'add README2.txt'

$ git log --oneline --graph --all --decorate
* 9f65aa9 (HEAD -> funcionalidade) add README2.txt
* ed9bf44 (master) add README1.txt
```

Utilizando branches (3)

```
git checkout master
git checkout -b bugx
echo 'bug' >> README1.txt
git commit -am 'BUGx'

$ git log --oneline --graph --all --decorate
* d5509e8 (HEAD -> bugx) BUGx
| * 9f65aa9 (funcionalidade) add README2.txt
|/
* ed9bf44 (master) add README1.txt
```

Utilizando branches (4)

```
git checkout master
echo 3 > README3.txt
git add README3.txt
git commit -m 'add README3.txt'

$ git log --oneline --graph --all --decorate
* e8ab14a (HEAD -> master) add README3.txt
| * d5509e8 (bugx) BUGx
|/
| * 9f65aa9 (funcionalidade) add README2.txt
|/
* ed9bf44 add README1.txt
```

Utilizando branches (5)

```
git checkout master
git merge bugx
```

```
$ git log --oneline --graph --all --decorate
*   f757761 (HEAD -> master) Merge branch 'bugx'
|\
| * d5509e8 (bugx) BUGx
* | e8ab14a add README3.txt
|/
| * 9f65aa9 (funcionalidade) add README2.txt
|/
* ed9bf44 add README1.txt
```

tag

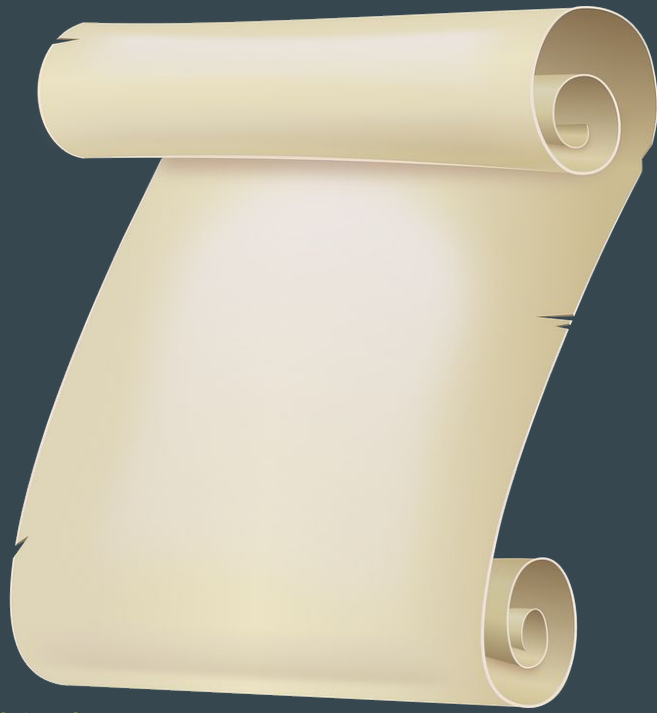
Criando uma tag:

```
git tag v0.0.1
```

Consultando as tags existentes:

```
git tag

git tag --list 'v0*'
```

Informações sobre commits



```
$ git log
...
commit f1bac8c5613fc928e85008c493e6b98b5321d6a5
Merge: 4876779 c6f97c8
Author: Jairo Gubler <jairo.gubler@gmail.com>
Date:   Sat Aug 18 16:11:35 2018 -0300

    Informações sobre git merge e git log
```

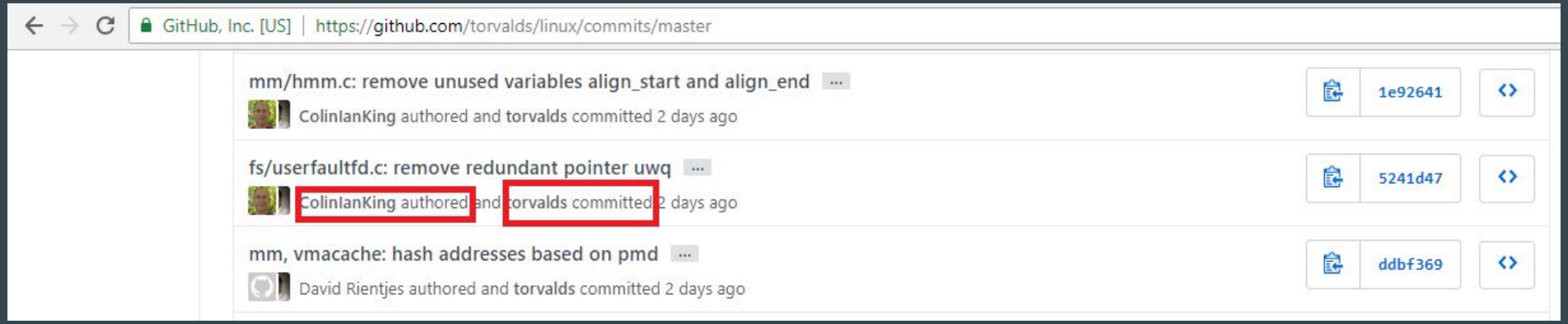
```
$ git cat-file -p f1bac8c5613fc928e85008c493e6b98b5321d6a5
tree ce56b2260a00ef65eb96806ca05c13bdbff1dd0d
parent 4876779c52e7dcfe8b2d30b176a3b6c6c30bbf09
parent c6f97c8ef5c38365c6b556ffd6cca7af76bb075c
author Jairo Gubler <jairo.gubler@gmail.com> 1534619495 -0300
committer Jairo Gubler <jairo.gubler@gmail.com> 1534619495 -0300

Informações sobre git merge e git log
```

Informações sobre commits

Author x commiter

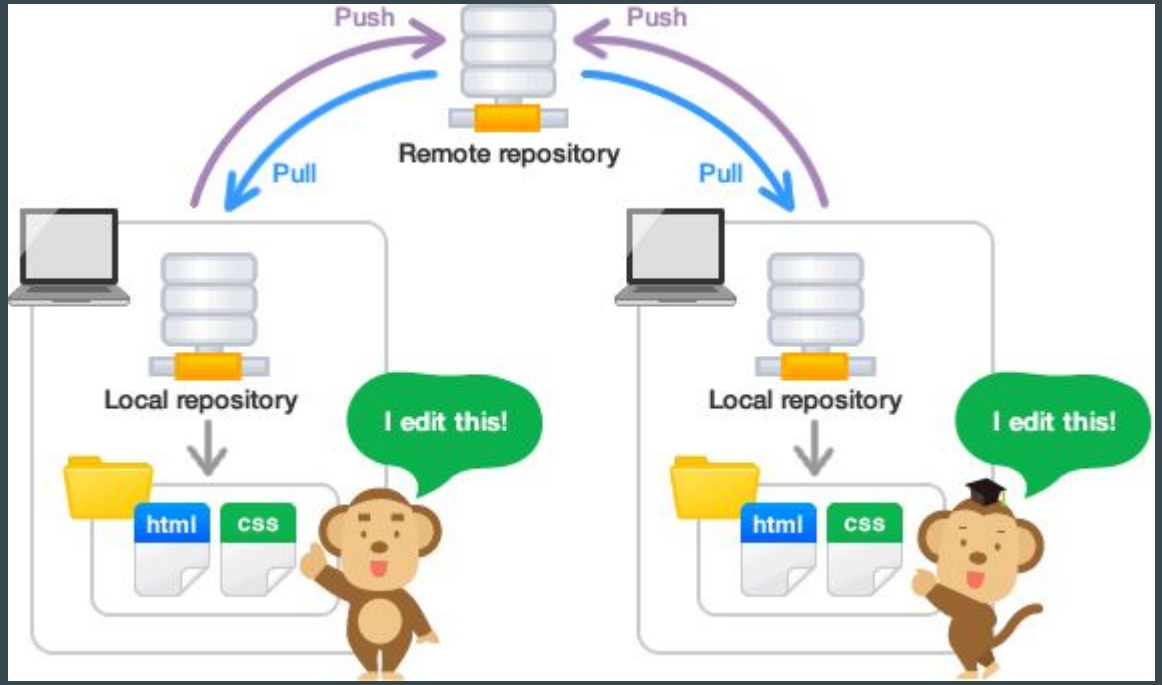
Author: pessoa que originalmente escreveu o patch.

Commiter: pessoa que aplicou o patch pela última vez no projeto.



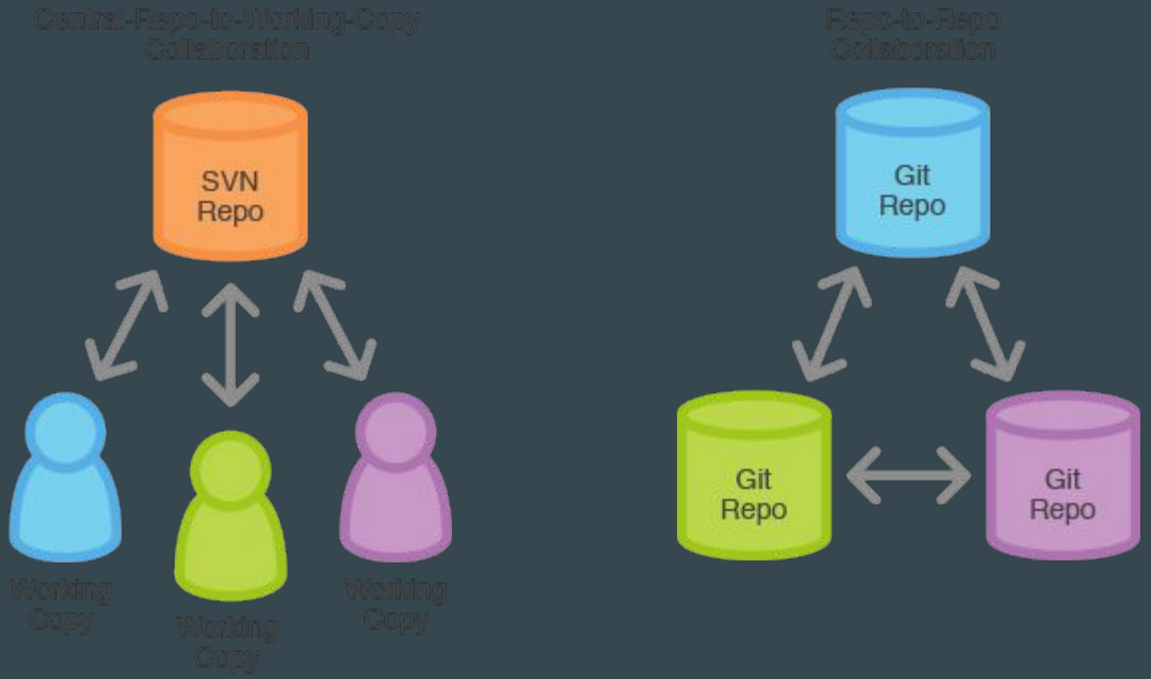
Estruturas internas: commit, parent, tree e blob

Repositórios Remotos



Clonando um repositório remoto

```
git clone https://github.com/nginx/nginx.git
```



git clone - etapas implícitas

`git clone https://github.com/nginx/nginx.git`

1. criação da pasta nginx e da estrutura da pasta nginx/.git (git init)
2. adição do repositório remoto (git remote add **origin** ...)
3. download do conteúdo do repositório para a pasta nginx/.git (git fetch)
4. cópia dos arquivos para o workspace (git checkout master)

Repositórios remotos

- https://github.com
- https://gitlab.com

Repositórios remotos - protocolos

- http/https
- ssh

Repositório remotos - protocolo http/https

Acesso através de usuário e senha do servidor (gitlab,github)

```
$ git config --global credential.helper
manager

$ git config --global credential.helper cache
```

https://git-scm.com/book/en/v2/Git-Tools-Credential-Storage

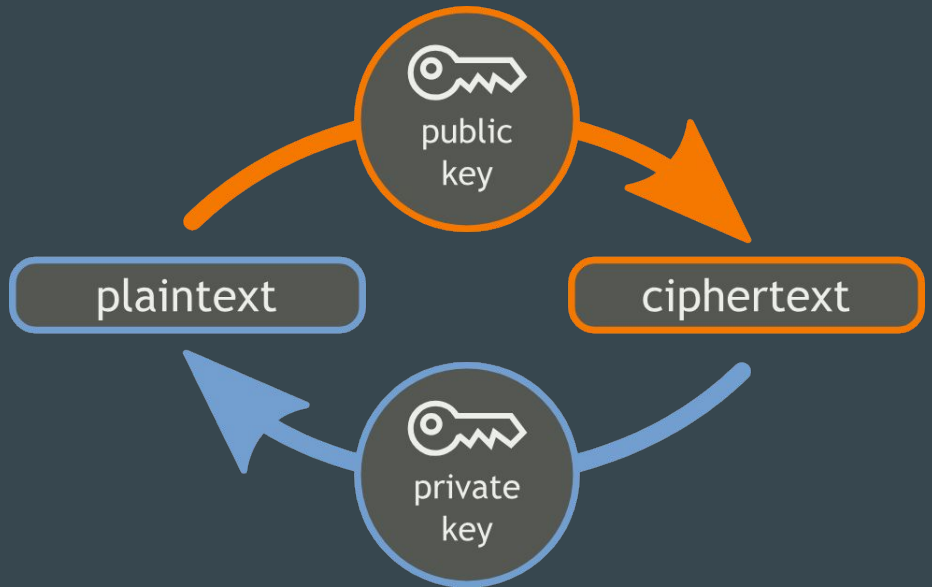
https://help.github.com/articles/caching-your-github-password-in-git/#platform-all

Repositório remotos - protocolo ssh

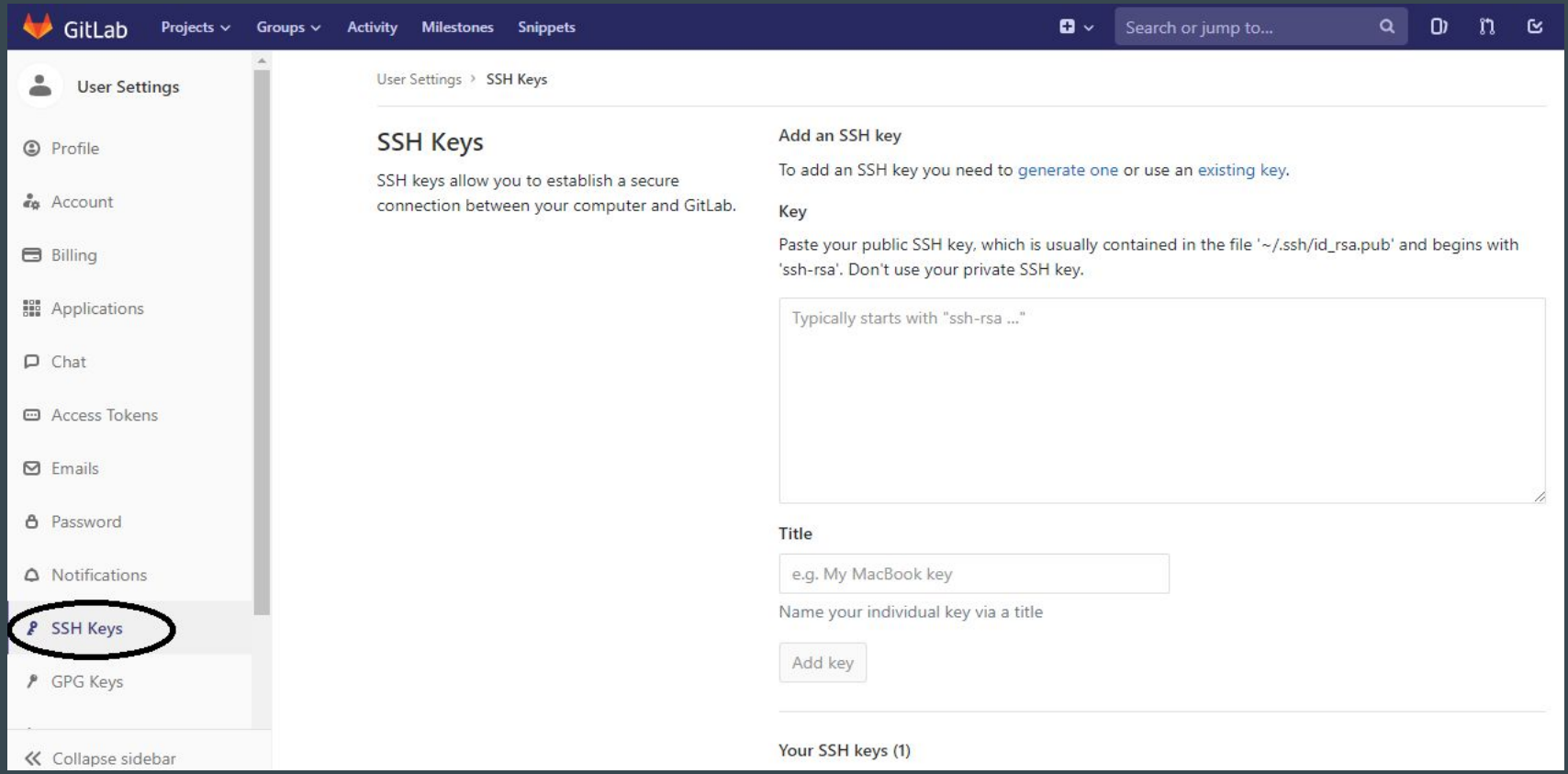
Utiliza uma chave ssh para acesso aos repositórios via protocolo ssh, configurável via interface WEB da ferramenta.

```
ssh-keygen -t rsa
```

```
cat ~/.ssh/id_rsa.pub
```



Repositório remotos - protocolo ssh

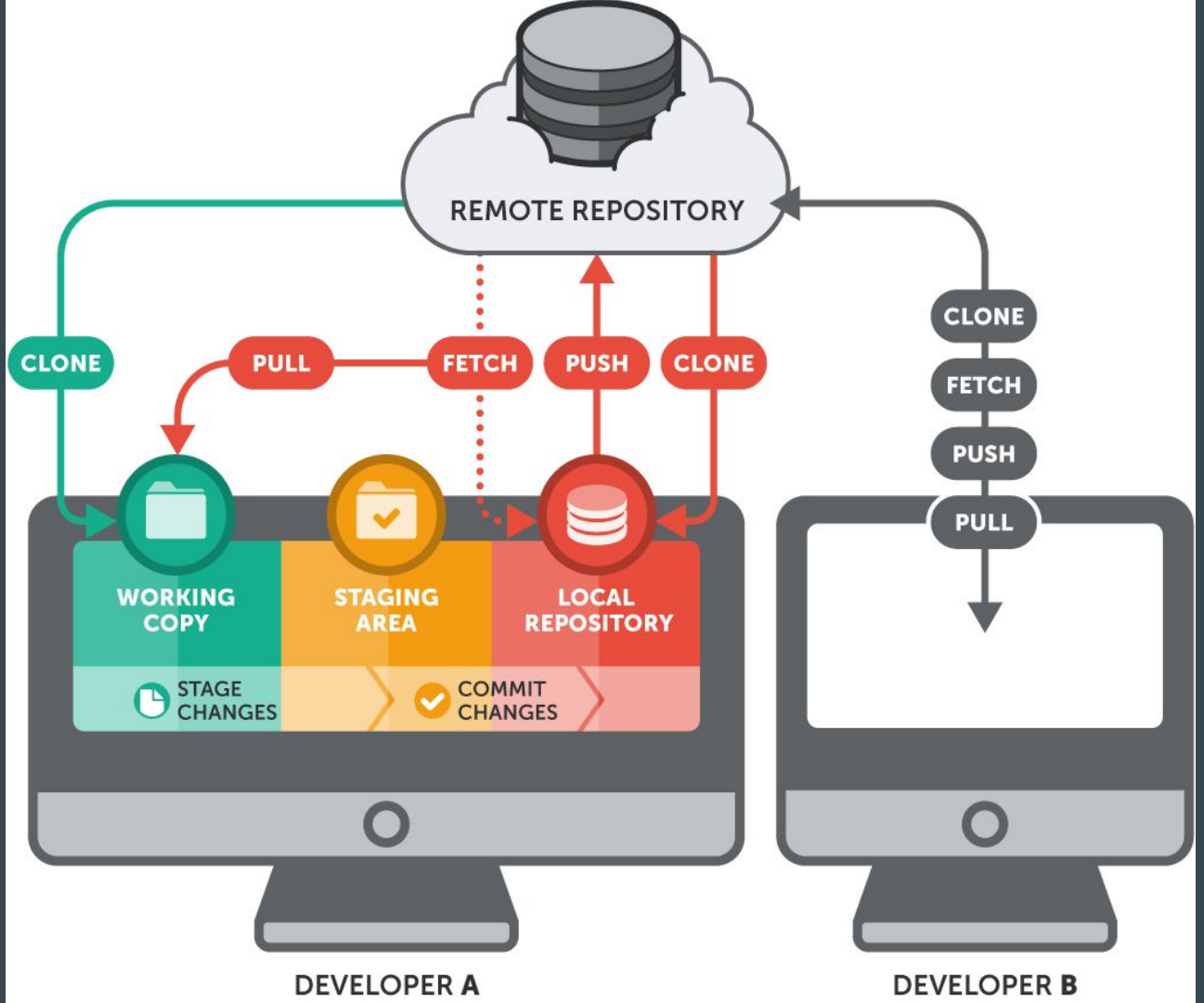


Propriedades do Projeto

- namespace (grupo ou usuário)
- controle de acesso (visibilidade, developers, owner)
- chaves de acesso (deploy keys)
- integrações (CI, webhooks)
- pull/merge request

Conceitos

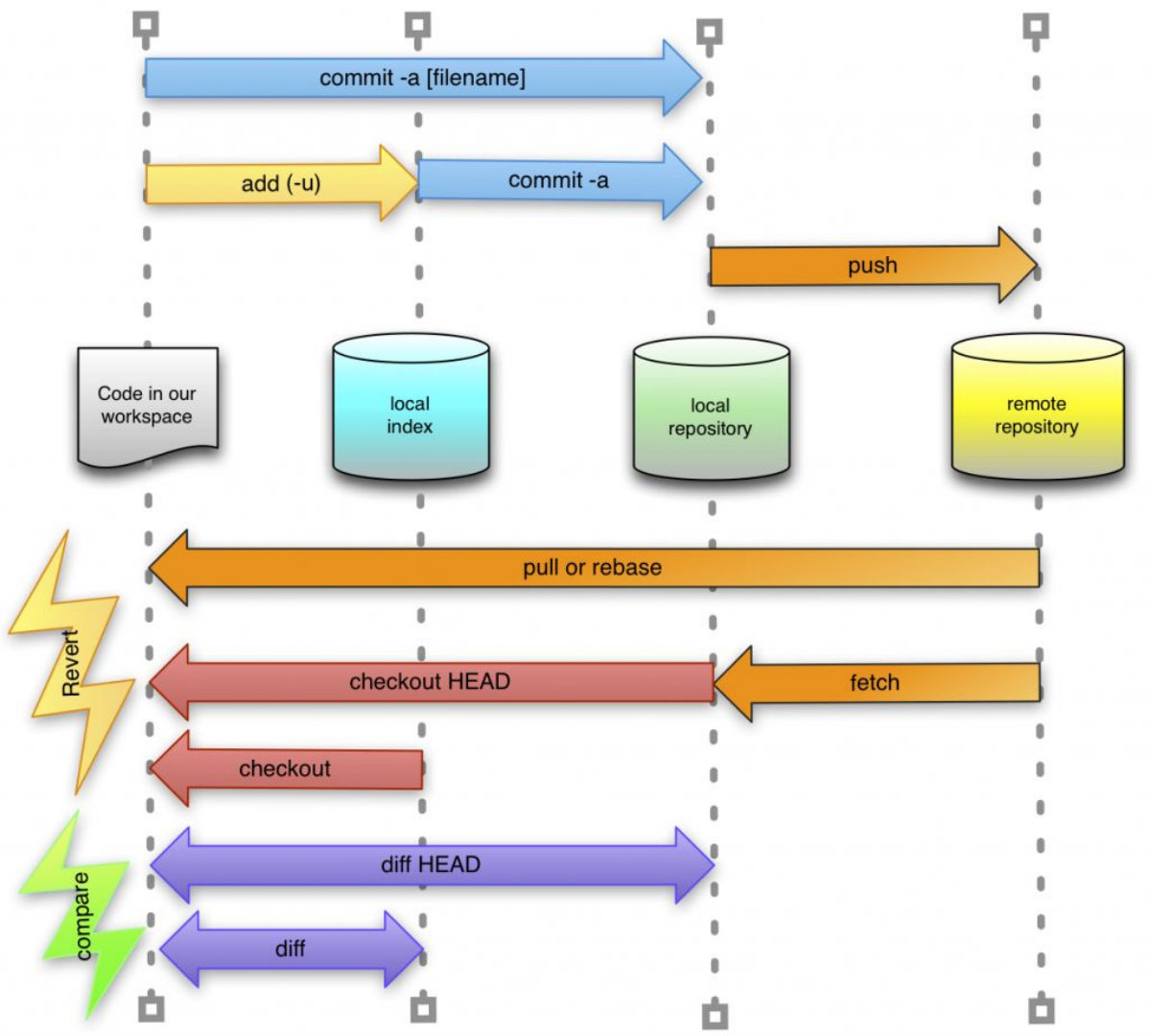
- master
- HEAD
- merge
- remote
- origin
- fetch
- push
- pull



git clone alterando nome do “remote”

```
$ git clone https://github.com/nginx/nginx.git -o github
Cloning into 'nginx'...
remote: Counting objects: 56932, done.
remote: Compressing objects: 100% (42/42), done.
remote: Total 56932 (delta 42), reused 57 (delta 36), pack-reused 56854
Receiving objects: 100% (56932/56932), 54.72 MiB | 747.00 KiB/s, done.
Resolving deltas: 100% (43752/43752), done.
$ cd nginx
$ git remote -v
github  https://github.com/nginx/nginx.git (fetch)
github  https://github.com/nginx/nginx.git (push)

$ git remote rename github origin
```

git pull, fetch, push



git push

Envia o branch para o repositório remoto no <branch>:

```
git push <remote> <branch>
```

Enviar tags para o repositório remoto:

```
git push <remote> --tags
```

git fetch

git fetch sincroniza o repositório local com commits ocorridos no repositório remoto.

git fetch não atualiza a área de trabalho e nem faz merge.

```
git fetch [repositorio-remoto] [branch/tag]
```

git pull

git pull sincroniza o repositório local com commits ocorridos no repositório remoto e faz merge com branch corrente.

```
git pull [repositorio-remoto] [branch/tag]
```

Fluxos de Trabalho

Fluxos de trabalho - Implementar nova funcionalidade

```
git checkout -b nova_funcionalidade master
vi …
git add …
git commit -m “nova funcionalidade implementada”
git checkout master
git pull origin master
git merge nova_funcionalidade
git branch -d nova_funcionalidade
git push origin master
```

Fluxos de trabalho - Implementar nova funcionalidade e bug

```
git checkout -b nova_funcionalidade master
vi ...
git stash

git checkout -b branch_v1.0.1 v1.0.0
vi x
git commit -am “correção do bug 1234”
git checkout master
git merge branch_v1.0.1
git checkout nova_funcionalidade
git stash pop
```

Fluxos de trabalho - dois repositórios

```
git clone git@gitlab.empresa.com.br/infra/janus-gateway.git

cd janus-gateway

git remote add github https://github.com/meetecho/janus-gateway.git

git fetch github --tags

git tag

git checkout -b nova_versao tag_local_antiga

git merge tag_remota_nova
```

Fluxo de Trabalho - transferir um commit para outro branch

```
git checkout master
vi teste.c
git commit -am 'teste.c'
```

Último commit deveria ser feito num branch

```
git branch desenvolvimento
git reset --hard HEAD~1
git checkout desenvolvimento
```

Fluxo de Trabalho - agrupar vários commit de um branch

```
git checkout -b funcionalidade-99 develop
… git commit … git commit … git commit …
```

Fazer o merge no branch develop

```
git checkout develop
git merge --squash funcionalidade-99
git add …
git commit
```

Fluxo de Trabalho - desfazer commits

Desfazendo commits e voltando arquivos alterados para a staging area

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

Desfazendo commits e devolvendo arquivos alterados para a working area

```
git reset <commit>
```

Desfazendo commits

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

Se necessário utilizar ***git reflog*** e *git reset --hard <commit>* para refazer commits.

Resumo de Comandos

Resumo de comandos - configuração de ambiente

```
git config --list
git config --global user.name "Fulano de Tal"
git config --global user.email "fulano.de.tal@dominio.com"
git config --global core.editor "vim"
git config --global merge.tool vimdiff
git config --global merge.tool meld
git config --global mergetool.meld.path "C:\Program Files
(x86)\Meld\Meld.exe"
git config --global diff.tool meld
git config --global difftool.meld.path "C:\Program Files
(x86)\Meld\Meld.exe"

cat ~/.gitconfig
```

Resumo de comandos - criar um repositório local

```
mkdir projeto
cd projeto
git init
```

Resumo de comandos - clonar um repositório

```
git clone <url> [pasta]

git clone
git://git.kernel.org/pub/scm/linux/kernel/git
/stable/linux-stable.git

git clone https://github.com/nginx/nginx.git
```

Resumo de comandos - arquivos na staging area

```
git add <arquivo ou pasta>

git rm <arquivo>

git rm -r <pasta>

git status
```

Resumo de comandos - commit

```
git commit

git commit -m "mensagem”

git commit -a

git commit -am "mensagem"

git commit --amend
```

Resumo de comandos - reset --hard

```
git checkout master
vi teste.c
git commit -am 'teste.c'
git branch desenvolvimento
git reset --hard HEAD~1
git checkout desenvolvimento
```

Resumo de comandos - branches

```
git branch

git branch -a

git branch desenvolvimento
```

Resumo de comandos - branches

```
git checkout <branch-name>

git checkout [-b <new-branch-name>] [<ref>]
```

git checkout desenvolvimento
git checkout v0.3.1
git checkout -b bug_1234 v0.3.1
git checkout -b nova_func desenvolvimento

Resumo de comandos - tags

```
git tag

git tag v1.0.1

git tag v1.0.0 <sha-1>

git tag -d v1.0.1

git tag -a -m "mensagem" <tag-name> [<sha-1>]
```

Resumo de comandos - remotes

```
git remote add <nome> <url>

git remote add origin
git@gitlab.empresa.com.br:jairo.gubler/projeto.git

git remote add github
git@github.com:jairogubler/projeto.git
```

Resumo de comandos - enviar repositório para remoto

```
git push <remote> <branch>

git push github master

git push origin master

git push origin desenvolvimento
```

Resumo de comandos - baixar novas versões (1)

```
git pull [options] [<repository> [<refspec>]]

git pull

git pull origin master
```

Resumo de comandos - baixar novas versões (2)

```
git fetch [<options>] [<repository> [<refspec>]]

git fetch github v0.3.1
```

Git cherry-pick

```
git cherry-pick <commit(s)>
```

Resumo de comandos - diff

```
git diff

git diff <branch-name>

git diff v0.2.3 v0.2.4

git diff --cached

git diff <commit> <commit> <filename>
```

Resumo de comandos - difftool

```
git config --global diff.tool meld

git config --global difftool.meld.path
"C:\Program Files (x86)\Meld\Meld.exe"

git difftool --dir-diff master

git difftool --dir-diff v0.2.3 v0.2.4
```

Resumo de comandos - log

```
git log
git log --pretty=oneline
git log --oneline
git log --oneline --decorate
git log --oneline --decorate --graph
git log --oneline --decorate --graph --all
git log --grep texto-qualquer
git config --global alias.logf "log --graph
--oneline --decorate"
git logf
```

git log

```
$ git log --all --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bol
* 658a84f4 - (HEAD -> master, origin/master, origin/branches/default, origin/HEAD) SSL: deprecated the "ssl
* 76be1ea9 - SSL: detect "listen ... ssl" without certificates (ticket #178). (7 days ago) <Maxim Dounin>
* 5d3a854e - Mail: fixed error message about missing ssl_certificate_key. (7 days ago) <Maxim Dounin>
* 8b4c7725 - Cache: fixed cache valid slot to reject incorrect statuses. (12 days ago) <Maxim Dounin>
* 59d806b5 - Version bump. (12 days ago) <Maxim Dounin>
| * 1165dc2b - (origin/branches/stable-1.14) release-1.14.0 tag (13 days ago) <Maxim Dounin>
| * a8fe224b - (tag: release-1.14.0) nginx-1.14.0-RELEASE (13 days ago) <Maxim Dounin>
| * 9bc4b9d1 - Stable branch. (13 days ago) <Maxim Dounin>
|/
* 53eae188 - release-1.13.12 tag (3 weeks ago) <Maxim Dounin>
* 4b9cc176 - (tag: release-1.13.12) nginx-1.13.12-RELEASE (3 weeks ago) <Maxim Dounin>
* f9e43a31 - Upstream: fixed u->conf->preserve_output (ticket #1519). (4 weeks ago) <Maxim Dounin>
* 923c9d5f - Version bump. (4 weeks ago) <Maxim Dounin>
* d627eeef - release-1.13.11 tag (4 weeks ago) <Maxim Dounin>
* ff70c2f1 - (tag: release-1.13.11) nginx-1.13.11-RELEASE (4 weeks ago) <Maxim Dounin>
* 703ca2b3 - Updated OpenSSL and PCRE used for win32 builds. (4 weeks ago) <Maxim Dounin>
* d90a75b7 - Upstream: fixed ngx_http_upstream_test_next() conditions. (4 weeks ago) <Maxim Dounin>
* cadc8ca3 - Core: fixed build, broken by 63e91f263a49. (4 weeks ago) <Vladimir Homutov>
* 55f08fc9 - Core: revised the PROXY protocol v2 code. (4 weeks ago) <Ruslan Ermilov>
* ecd6e243 - Core: style. (5 weeks ago) <Vladimir Homutov>
* 9207cc84 - Core: added processing of version 2 of the PROXY protocol. (6 weeks ago) <Vladimir Homutov>
* f39d5e8b - Stream: set action before each recv/send while proxying. (6 weeks ago) <Roman Arutyunyan>
* 0ac57648 - gRPC: fixed possible sign extension of error and setting_value. (6 weeks ago) <Maxim Dounin>
```

Resumo de comandos - .gitignore

```
cat .gitignore
```

https://git-scm.com/docs/gitignore

Resumo de comandos - submodule

Semelhante ao externals do SVN:

Adicionar um submódulo a um projeto:

```
git submodule add $repository $sub_dir
```

```
git submodule add -b build-64bits
git@gitlab.empresa.com.br:crazylifeteam/media
-streaming-server.git media-streaming-server
```

Resumo de comandos - submodule

Baixar submódulos de um projeto recém clonado ao atualizar alterações de submódulos:

```
git submodule init
git submodule update
```

https://git-scm.com/book/pt-br/v1/Ferramentas-do-Git-Subm%C3%B3dulos

Migração de projetos do svn para o git

Sem considerar o histórico de revisões do svn:

```
svn export http://repositorio/path-do-projeto/trunk projeto
cd projeto
git init
git add * (cuidado para não incluir a pasta de controle do svn)
git commit
git remote add origin git@...  (criar o projeto antes)
git push -u origin master
```

Migração de projetos do svn para o git

Considerando o histórico de revisões:

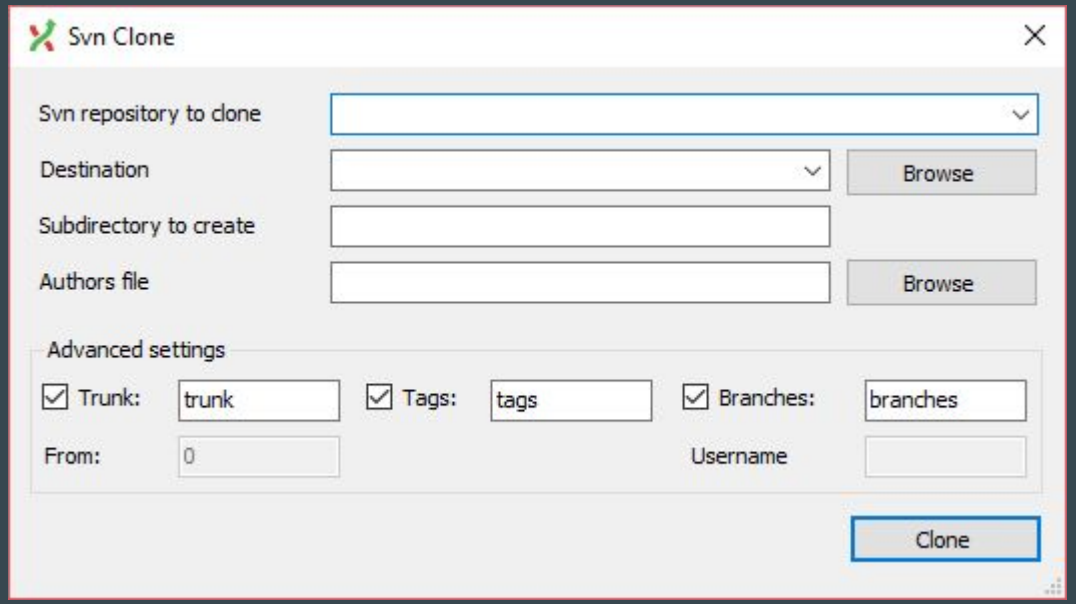
- instalar git-svn
- criar projeto no gitlab
- criar arquivo authors.txt
  (usuario-svn = Nome-Git <e-mail-git>)

Migração de projetos do svn para o git

```
git svn clone http://repositorio/path-do-projeto -T trunk -b
branches -t tags -A authors.txt --prefix svn/

cd path-do-projeto
git remote add origin git@xxx.git
git branch -a
git push -u origin --all
```

Migração de projetos do svn para o git

`https://github.com/gitextensions/gitextensions`



Referências

- https://git-scm.com/book/en/v2
- https://git-scm.com/book/pt-br/v2
- https://www.youtube.com/playlist?list=PLQCmSnNFVYnRdgxOC_ufH58NxImM6VYd1 (Rodrigo Branas, git)