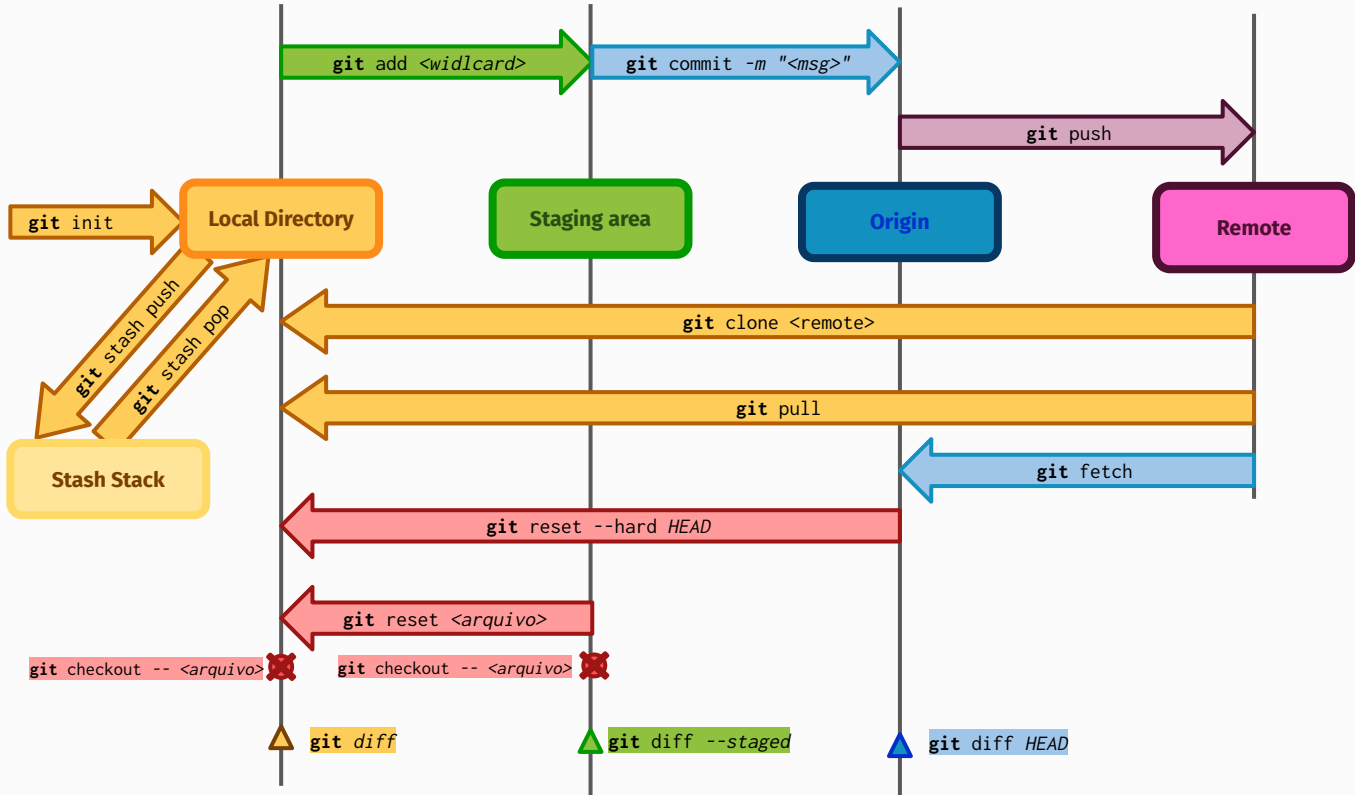


# CheatSheet - Git e GitHub

[github.com/ALREstevam](https://github.com/ALREstevam)

## Exemplo de Git Workflow

Baseado em <https://blog.osteele.com/2008/05/my-git-workflow/> e em <https://stackoverflow.com/a/3690796>



## Notação

~\$ comando ou simplesmente comando para entrada de comando no terminal como o usuário padrão  
# comando para entrada de comando no terminal como o usuário root  
# texto para comentário  
> texto ou simplesmente texto para saída de um comando executado no terminal  
Alguns argumentos possuem duas versões, pesquise e use a que preferir (-m e --message, por exemplo)

Nota sobre os prefixos e cores utilizados nos comandos de terminal deste arquivo

## Setup

```
sudo apt-get update
sudo apt-get install git -y
git --version
sudo apt-get install gitk -y
gitk
```

Instalar git e o gitk (interface gráfica simples) no Linux via apt-get e conferir a instalação pedindo a versão do git e rodando o gitk

```
git config --global user.name "Usuário no GitHub"
git config --global user.email "emaildeusuario@dogithub.com"
```

Adicionar a conta do GitHub ao git

```
git remote add origin <remote>
# Exemplo:
git remote add origin https://github.com/user/repo.git
```

Conectar origin e remote no diretório atual

```
git init .
```

Criar um novo repositório no diretório atual

```
git clone <remote>
# Exemplo:
git clone https://github.com/user/repo.git
```


Obter os arquivos no remote e fazer merge com as alterações locais



Comandos de transporte de dados


**git add <wildcard>**

Adicionar arquivos na staging area




**git commit -a -m "<msg>" -m "<msg>"**

Adicionar arquivos na staging area e fazer um commit




**git push**

Push dos commits locais para o remote




**git pull**

Obter os arquivos no remote e fazer merge com as alterações locais




**git reset -HEAD**

Desfazer último commit local




**git reset <arquivo>**

Remover arquivos da staging area



**git fetch**

Obter os arquivos no remote mas não fazer merge com as alterações locais



**git checkout -- <arquivo>**

Desfazer alterações no diretório de trabalho ou na staging area - os arquivos ficarão como os do último commit

Mostrar informação (com exemplos de possíveis saídas)

Obs.: para saídas muito longas navegue com as setas do teclado e saia com a tecla “q”

```
~$ git status
> On branch <nome-branch>
> Changes to be committed:
>   (use "git reset HEAD <file>..." to unstage)
>
>       new file:   file_on_staging_area.txt
>
> Changes not staged for commit:
>   (use "git add/rm <file>..." to update what will be
>   committed)
>   (use "git checkout -- <file>..." to discard changes in
>   working directory)
>
>       deleted:    file_to_be_deleted_after_add_area.txt
>
> Untracked files:
>   (use "git add <file>..." to include in what will be
>   committed)
>
>       file_on_local_directory.txt
>
```

Mostra o estado atual do diretório de trabalho: arquivos na staging area, arquivos não rastreados, arquivos deletados e arquivos que estavam na staging area, mas foram deletados

```
~$ git log
> commit 6f6098fc0ea9cbc6545d02300426434817ba8927 (HEAD ->
> master)
> Author: ALREstevam <a166348@dac.unicamp.br>
> Date:   Mon Mar 9 21:55:10 2020 -0300
>
>       Adicionando o segundo arquivo
>
>       Este é um comentário
>
> commit 1e12c9c48ed6a58977483331ede02511fd0002cd
> Author: ALREstevam <a166348@dac.unicamp.br>
> Date:   Mon Mar 9 21:53:23 2020 -0300
>
>       adicionando o primeiro arquivo
```

Lista dos commits até o estado atual do projeto

```
~$ git log --pretty=oneline
> 6f6098fc0ea9cbc6545d02300426434817ba8927 (HEAD -> master)
> Adicionando o segundo arquivo
> 1e12c9c48ed6a58977483331ede02511fd0002cd adicionando o
> primeiro arquivo
```

Forma rápida de listar apenas as mensagens e hashes dos commits

```
~$ git blame <arquivo>
# [...]
> 9faca74a (ALREstevam 2020-03-09 22:04:18 -0300 1) adicionando
> uma linha
> 9faca74a (ALREstevam 2020-03-09 22:04:18 -0300 2) adicionando
> outra linha
```

Mostrar quem, quando e o que foi alterado

```
~$ git log -p <arquivo>
# [...]
> commit 1e12c9c48ed6a58977483331ede02511fd0002cd
> Author: ALREstevam <a166348@dac.unicamp.br>
> Date:   Mon Mar 9 21:53:23 2020 -0300
>
>       adicionando o primeiro arquivo
>
> diff --git a/primeiro_arquivo.txt
> b/primeiro_arquivo.txt
> new file mode 100644
> index 0000000..0b80307
> --- /dev/null
> +++ b/primeiro_arquivo.txt
> @@ -0,0 +1 @@
> +adicionando uma linha
> \ No newline at end of file
```

Forma rápida de listar apenas as mensagens e hashes dos commits

**gitk**

Inicia a interface gráfica gitk - bastante útil para consultar informações e visualizar o commits e branches em um estrutura de árvore

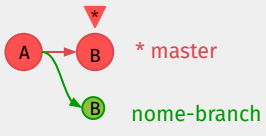
```
# use diff <arquivo> para especificar o arquivo
~$ git diff # Arquivos não rastreados
~$ git diff --staged # Arquivos na staging area
~$ git blame # Arquivos não rastreados e na staging area
> diff --git a/primeiro_arquivo.txt b/primeiro_arquivo.txt
> index 9287b54..7633552 100644
> --- a/primeiro_arquivo.txt
> +++ b/primeiro_arquivo.txt
> @@ -1,2 +1,3 @@
> Apagando as linhas e trocando por outra coisa
> -Números: 1, 2, 3
> \ No newline at end of file
> +Números: 1, 2, 3, 4
> +Letras: A, B, C
```

Mostrando as alterações nos arquivos em comparação com o último commit

Branches e tags

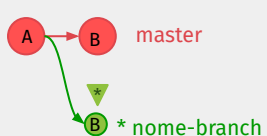
```
git branch <nome-branch>
```

Criar uma branch



```
git checkout <nome-branch>
```

Mudar branch no diretório de trabalho




```
git branch # Branches locais
git branch -a # '--all' todas as branches (origin e remote)
git branch -r # '--remote' branches no remote
git branch -av # branches locais e remotas com o título e hash do último commit
```

Listar branches locais, no remote ou ambas

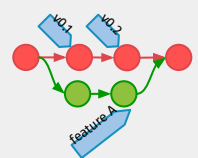
```
git branch -d <nome-branch>
```

Deletar uma branch




```
git tag
```

Lista tags do repositório



```
git tag -a v0.0 -m "<comentário>" <hash do commit>
# No último commit
git tag -a v0.1 -m "<comentário>"
```

Adicionar uma nova tag no último commit ou em algum commit específico. "v0.0" é uma máscara comum para tags, geralmente utilizadas para indicar a versão do software naquele commit



```
git show v0.0
```

Mostra informações sobre o commit com a tag "v0.0". Mude para a tag de escolha, use git tag para listar todas

```
git stash push # Ou somente 'git stash'
# Checkout para outro branch, alterações commit, checkout de volta para o branch atual
git stash pop
```

Limpando o working directory sem perder as alterações na staging area e os arquivos não rastreados - útil quando você precisa mudar de branch para fazer um hotfix, por exemplo, e não quer fazer commit das alterações em seu branch atual, mas também não quer perdê-las

Erros comuns e suas soluções

Baseado em <https://dangitgit.com/>

```
git commit --amend -m "<nova mensagem>"
```

Alterar a mensagem do último commit

**ATENÇÃO:** não faça amend e commits públicos (no remote) apenas nos que ainda estão locais (em origin)

```
# Fazer a alteração ...
git add <wildcard>
git commit --amend --no-edit
```

Fazer uma pequena alteração no último commit sem ter de fazer um novo

**ATENÇÃO:** não faça amend e commits públicos (no remote) apenas nos que ainda estão locais (em origin)

```
# Fazer a alteração ...
git branch <nome-nova-branch>
# Remover último commit da master
git reset HEAD~ --hard
git checkout <nome-nova-branch>
# Commit no novo branch
```

Commit feito na master mas deveria ser em uma nova branch

```
# desfazer o último commit mas deixar as alterações disponíveis
git reset HEAD~ --soft
# salvar as alterações temporariamente
git stash
# ir para a branch correta
git checkout <branch-correta>
# recuperar as mudanças salvas temporariamente
git stash pop
# marcar os arquivos para commit (colocar na staging area)
git add <wildcard>
git commit -m "<mensagem>"
```

Commit na branch errada (fora a master)

```
git checkout <branch-correta>
# Colocar o último commit da master como o último commit desta branch
git cherry-pick master
git checkout master
# Remover o último commit da master
git reset HEAD~ --hard
```

Commit feito na master mas deveria ser em uma nova branch (resolvido com cherry-pick)

```
git log # encontre a hash do commit com problemas e.g. 9fceb02
git revert 9fceb02
```

Commit que reverte as alterações de outro

```
git reset --hard
```

Desfazer alterações em todos os arquivos locais

```
git reset # Remover todos
git reset <arquivo> # Remover arquivo específico
```

Remover arquivos da staging area (que passaram pelo git add)

```
git checkout -- <arquivo>
```

Desfazer alterações não commitadas em um arquivo

Diretórios e arquivos

```
cd .. # Entra no diretório imediatamente superior
cd ~ # Entra em /home/<seu-usuário>
cd / # Entra no diretório raiz
```

Atalhos comuns na navegação

```
cd <diretório>
cd ./<diretório>
```

Entrar em um diretório a partir do atual

```
cd /<diretório>
```

Entrar em um diretório a partir do diretório raiz

```
mkdir <nome-dir>
```

Criar um diretório

```
ls # Imprime os arquivos e subdiretórios
ls -la # Imprime os arquivos e subdiretórios (ocultos inclusive) com mais informações
ls -alh # Imprime os arquivos e subdiretórios (ocultos inclusive) com mais informações e tamanhos legíveis
```

Imprimir arquivos e subdiretórios

```
~$ pwd
> /home/andre/Documents/git
```

Imprimir na tela o diretório atual

```
rm <arquivo>
rm -rf <diretório>
```

Remover um arquivo ou um diretório

```
grep "<texto>" <arquivo>
```

Imprimir as linhas de <arquivo> que contém a string <texto>

```
file <arquivo>
```

Informações sobre um arquivo

```
more <arquivo>
```

Imprime o conteúdo de um arquivo

```
history
```

Mostrar o histórico recente de comandos ("q" para fechar)

```
echo "<texto>" > <arquivo>
```

Forma rápida de criar um arquivo contendo <texto>

```
cp <arquivo_origem> <arquivo_destino>
rm -r <diretório_origem> <diretório_destino>
```

Copiar arquivos ou diretórios

```
nano <arquivo>
[ ctrl + x ] > [ n ] # Sair sem salvar
[ ctrl + o ] > [ enter ] > [ ctrl + x ] #
Escrever alterações e sair
```

Simple editor de arquivo (o vim é um editor melhor, mas mais difícil de aprender)

```
ls <wildcard>
ls * # Qualquer string com qualquer quantidade de caracteres
ls *.c # Qualquer arquivo ou diretório com a extensão '.c'
ls *maçã* # Qualquer arquivo ou diretório que contenha a palavra `maça` em seu nome
ls ??? # Qualquer arquivo ou diretório em que o nome possui exatamente três caracteres
```

Wildcards para seleção de arquivos

```
Se você acidentalmente sair do nano, execute-o novamente no arquivo para recuperar a edição
```

Dicas e outros comandos úteis

```
[Esc] # Alternar entres os modos de edição e de inserção de comandos
:q! # Sair do vim e não salvar nada (dica suprema)
:wq # Sair do vim e salvar as alterações
```

Dicas para o editor Vim - fazer um commit sem a opção -m abrirá o Vim para que você escreva a mensagem de commit

```
gedit <arquivo>
```

Abrir o gedit em um arquivo

```
subl .
subl <arquivo>
```

Abrir o Sublime text no diretório atual ou em um arquivo

```
code .
code <arquivo>
```

Abrir o Visual Studio code no diretório atual ou em um arquivo

```
# 0 comando base será
git log --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<an>%Creset' --abbrev-commit --date=relative

# 0 salvaremos em um alias, assim poderemos acessá-lo apenas digitando `git lg`

git config --global alias.lg "log --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<an>%Creset' --abbrev-commit --date=relative"

git lg # Lista de commits com hash parcial, branch, título, há quanto tempo foi o commit e quem o fez
git lg -p # As informações anteriores e quais linhas foram alteradas ou adicionadas

~$ git lg
# [...]
> * 9faca74 - Alterando o primeiro arquivo de exemplo (3 hours ago) <ALREstavam>
> * 6f6098f - Adicionando o segundo arquivo de exemplo (10 hours ago) <ALREstavam>
> * 1e12c9c - Adicionando o terceiro arquivo de exemplo (15 hours ago) <ALREstavam>
```

Um git log melhorado  
(extraído de <https://coderwall.com/p/euwpig/a-better-git-log>)

Outras fontes: <https://metring.com.br/git-desfazendo-mudancas-locais>, [www.cheat-sheets.org/saved-copy/git-cheat-sheet.pdf](http://www.cheat-sheets.org/saved-copy/git-cheat-sheet.pdf), <https://git-scm.com/book/en/v2>