

# Git Cheat Sheet

<http://git.or.cz/>

Versão em Português <http://rgou.net/cheatsheets/>

Lembre-se: `git command --help`

Configuração Global do Git é armazenada em `$HOME/.gitconfig` (`git config --help`)

## Criar

De dados existentes

```
cd ~/projects/myproject
git init
git add .
```

De um repositório existente

```
git clone ~/existing/repo ~/new/repo
git clone git://host.org/project.git
git clone ssh://you@host.org/proj.git
```

## Mostrar (show)

Arquivos modificados no diretório atual

```
git status
```

Modificações para arquivos rastreados

```
git diff
```

O que mudou entre \$ID1 e \$ID2

```
git diff $id1 $id2
```

Histórico de modificações

```
git log
```

Historico de modificações para arquivo com diffs

```
git log -p $file $dir/ec/tory/
```

Quem modificou o que e quando um arquivo

```
git blame $file
```

Um commit identificado por \$ID

```
git show $id
```

Um arquivo específico de um \$ID específico

```
git show $id:$file
```

Todos ramos (branches) locais

```
git branch
```

(asterisco '\*' marca o ramo (branch) atual)

## Conceitos

### Básico do Git

```
master : ramo (branch) de desenvolvimento padrão
origin  : repositório upstream padrão
HEAD   : ramo (branch) atual
HEAD~  : pai (parent) de HEAD
HEAD~4 : o tetravô de HEAD
```

### Reverter

Retornar ao último status de commit

```
git reset --hard
```

⚠️ você não pode desfazer um hard reset

Reverter o último commit

```
git revert HEAD
```

 Cria um novo commit

Revert um commit específico

```
git revert $id
```

 Cria um novo commit

Corrige o último last commit

```
git commit -a --amend
```

(após editar arquivos quebrados)

Checkout a versão \$id de um arquivo

```
git checkout $id $file
```

### Ramo (Branch)

Mudar parao ramo \$id

```
git checkout $id
```

Mesclar (merge) branch1 no branch2

```
git checkout $branch2
git merge branch1
```

Criar um ramo chamado \$branch

```
git branch $branch
```

Criar ramo \$new\_branch baseado

```
no ramo $other e mudar para ele
```

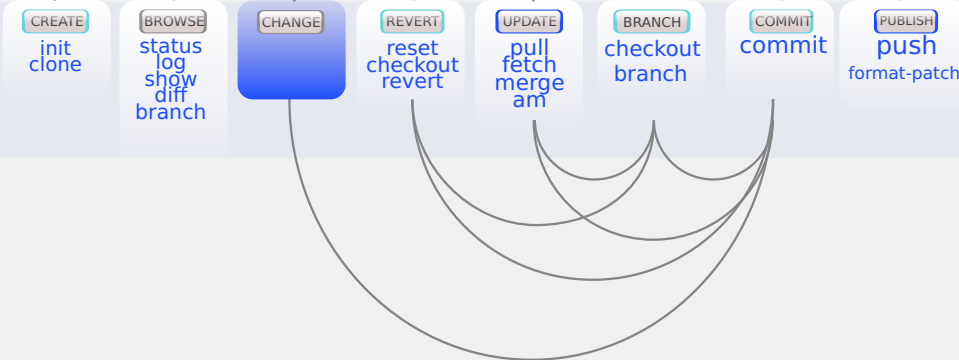
```
git checkout -b $new_branch $other
```

Excluir ramo \$branch

```
git branch -d $branch
```

## Sequência de Comandos

as curvas indicam que o comando à direita é geralmente executado após o comando à esquerda. Isto dá uma idéia do fluxo de comandos usualmente feitos com o Git.



## Criar

## Conceitos

## Atualizar (update)

Recuperar últimas mudanças da origem

```
git fetch
```

(mas isto não as mescla (merge)).

Recupera e aplica as últimas

modificações da origem

```
git pull
```

(faz um "fetch" seguido de um "merge")

Aplica um patch que alguém te enviou

```
git am -3 patch.mbox
```

(em caso de conflito, resolve e usa `git am --resolved`)

## Publicar (publish)

Faz um Commit de todas mudanças locais

```
git commit -a
```

Prepara um patch para outros desenvolvedores

```
git format-patch origin
```

Envia modificações para a origem

```
git push
```

Marca uma versão / marco (milestone)

```
git tag v1.0
```

## Useful Commands

Finding regressions

```
git bisect start (to start)
git bisect good $id ($id is the last working version)
git bisect bad $id ($id is a broken version)
```

```
git bisect bad/good (to mark it as bad or good)
git bisect visualize (to launch gitk and mark it)
git bisect reset (once you're done)
```

Check for errors and cleanup repository

```
git fsck
git gc --prune
```

Search working directory for foo()

```
git grep "foo()"
```

## Resolve Merge Conflicts

To view the merge conflicts

```
git diff (complete conflict diff)
git diff --base $file (against base file)
git diff --ours $file (against your changes)
git diff --theirs $file (against other changes)
```

To discard conflicting patch

```
git reset --hard
git rebase --skip
```

After resolving conflicts, merge with

```
git add $conflicting_file (do for all resolved files)
git rebase --continue
```

## Cheat Sheet Notation

\$id : notação usada nesta folha para representar um id de commit, id de branch ou um nome de tag  
\$file : nome arbitrário de arquivo  
\$branch : nome arbitrário de branch