Introdução ao Git

Sistema de controle de versão distribuído

Estagiários PAE: Brauner Oliveira / Lina Rodriguez
Prof.ª Elisa Yumi Nakagawa
SSC 0527/0620 - Engenharia de Software
1º Semestre de 2015

Agenda

- Fundamentos
- Instalação
- Principais comandos
- Material complementar

Fundamentos

Fundamentos

Gerenciamento de Configuração → Controle de Versão → Ferramentas → Git

Git controla a versão de itens de configuração de produto de software e produto de desenvolvimento de software, mas seu potencial está em itens de texto simples (código-fonte, arquivos de configuração)

Criado por Linus Torvalds para versionar o código-fonte do Kernel do Linux, que é mantido por colaboradores ao redor do mundo

Outros sistema de controle de versão: CVS, SVN, Mercurial

Fundamentos

Mudanças nos artefatos durante o processo de desenvolvimento (e.g., adição de novos requisitos, bug fixes, anotações e alteração dos diagramas UML)

Artefatos são geralmente representados por arquivos (binários ou textuais)

Git controla as alterações feitas durante o processo: adição, remoção e modificação de arquivos (hash sha-1)

Mantém histórico de todas as alterações (repositório do Linux: ~1 GB)

Fundamentos

Trabalho colaborativo

Modelo convencional (sem colaboração) → Sobrescrita, *File Locking*

Edição paralela de um mesmo arquivo

Fulano A trabalha na função X
Fulano B trabalha na função Y
Cada um faz as alterações localmente
Por fim, as duas mudanças são mescladas

Fundamentos

Social coding

Ferramentas construídas com base no git
Extendem algumas funcionalidades
Comunidade Open-Source

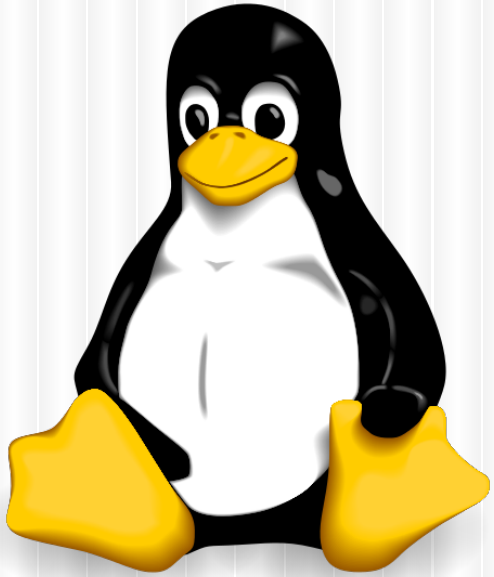
GitHub
BitBucket

Atlassian Bitbucket

Fundamentos

Quem usa git?

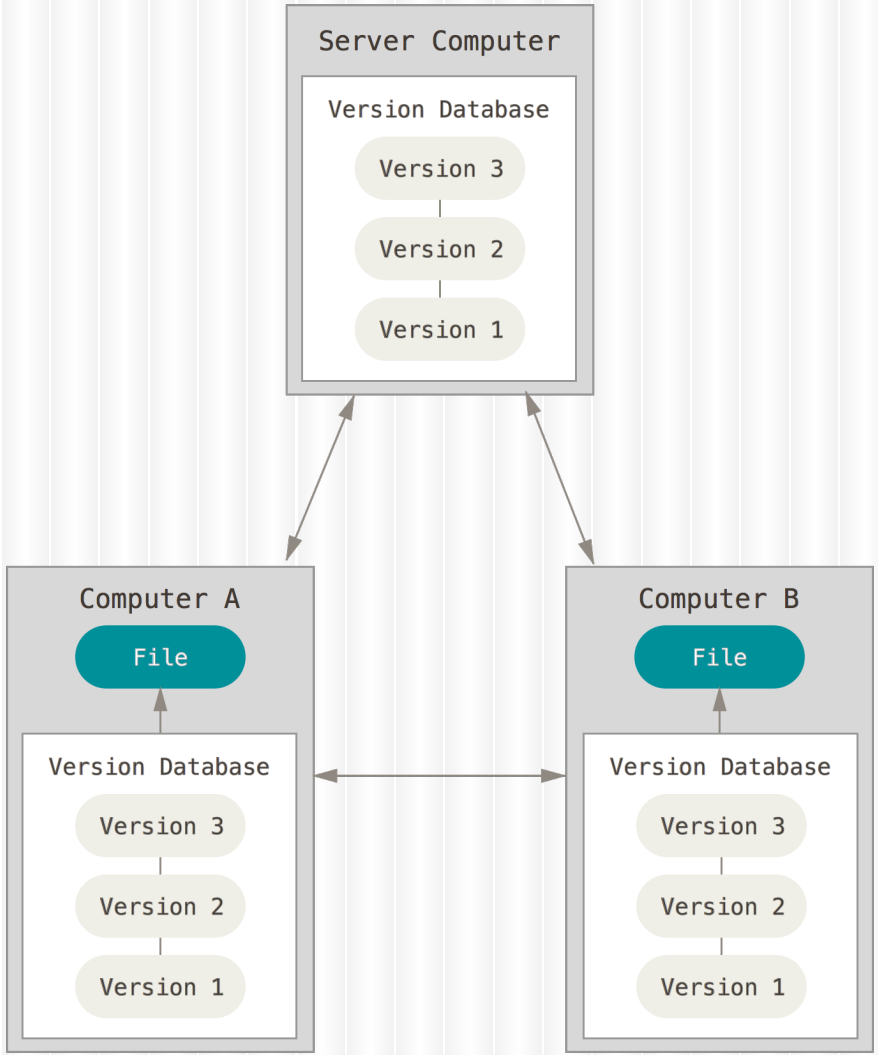
Google™

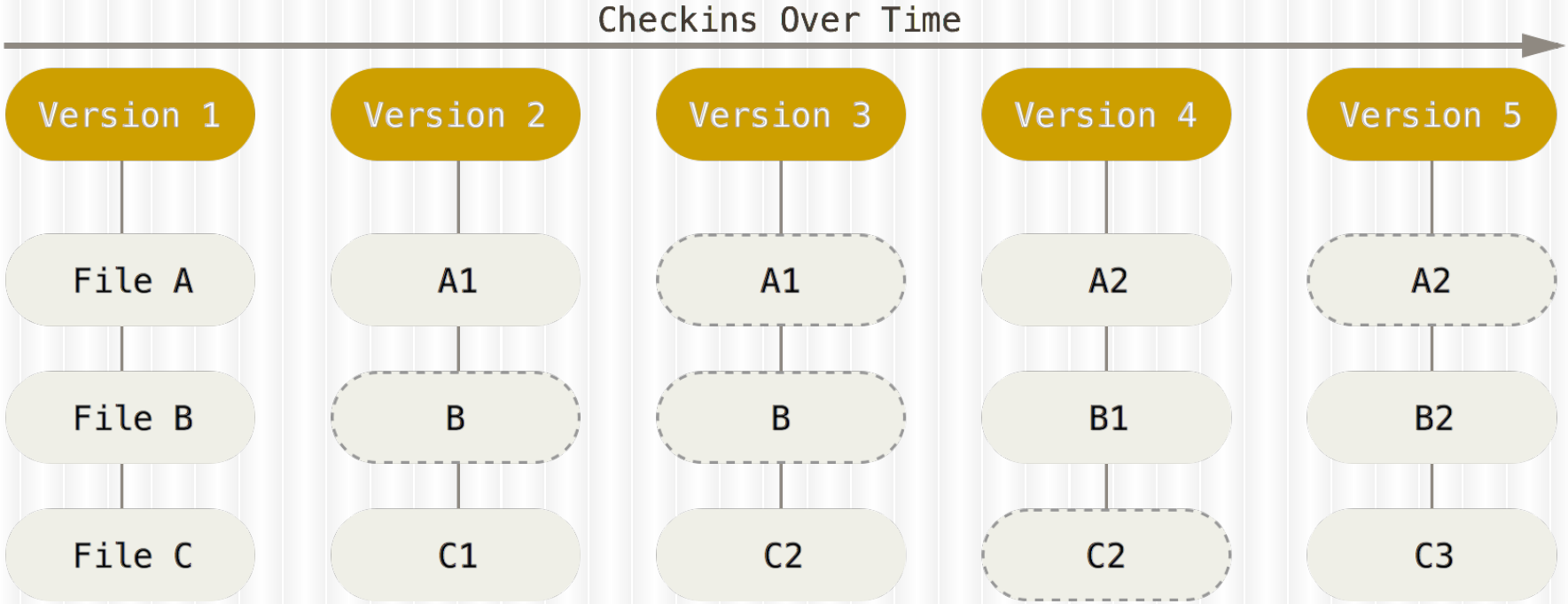


facebook

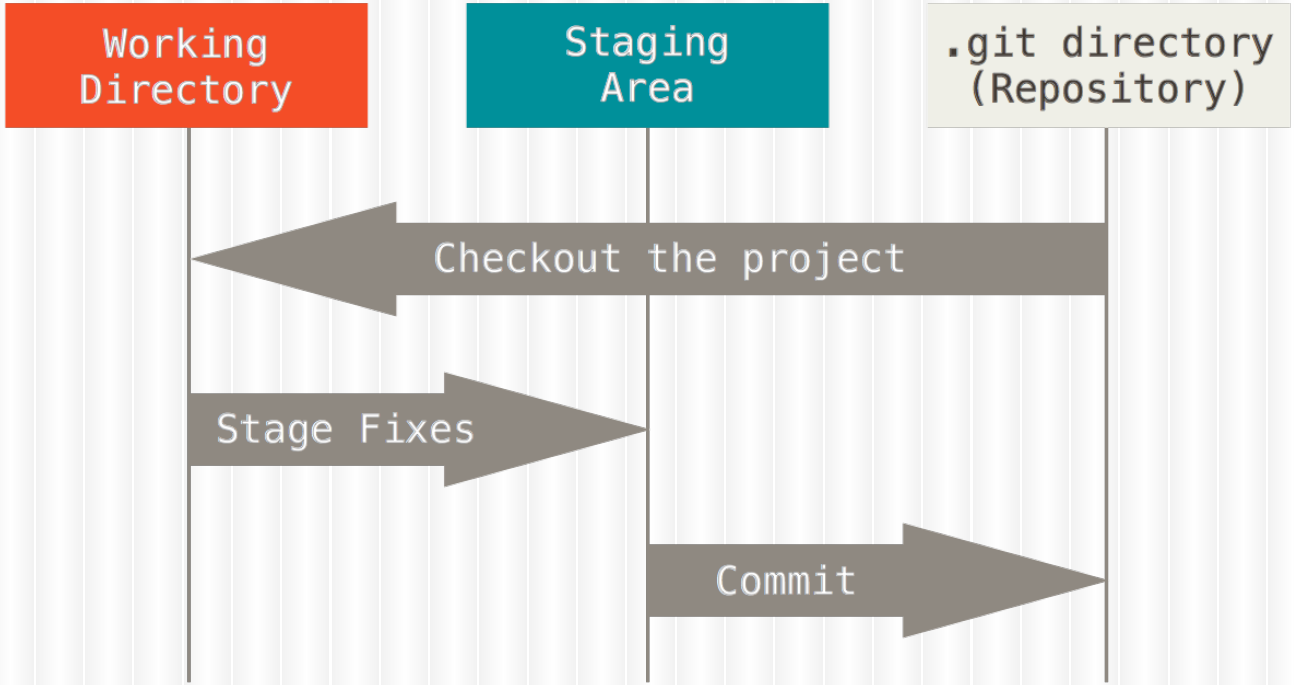
Microsoft

Arquitetura de Sistemas de Controle de Versão Distribuídos

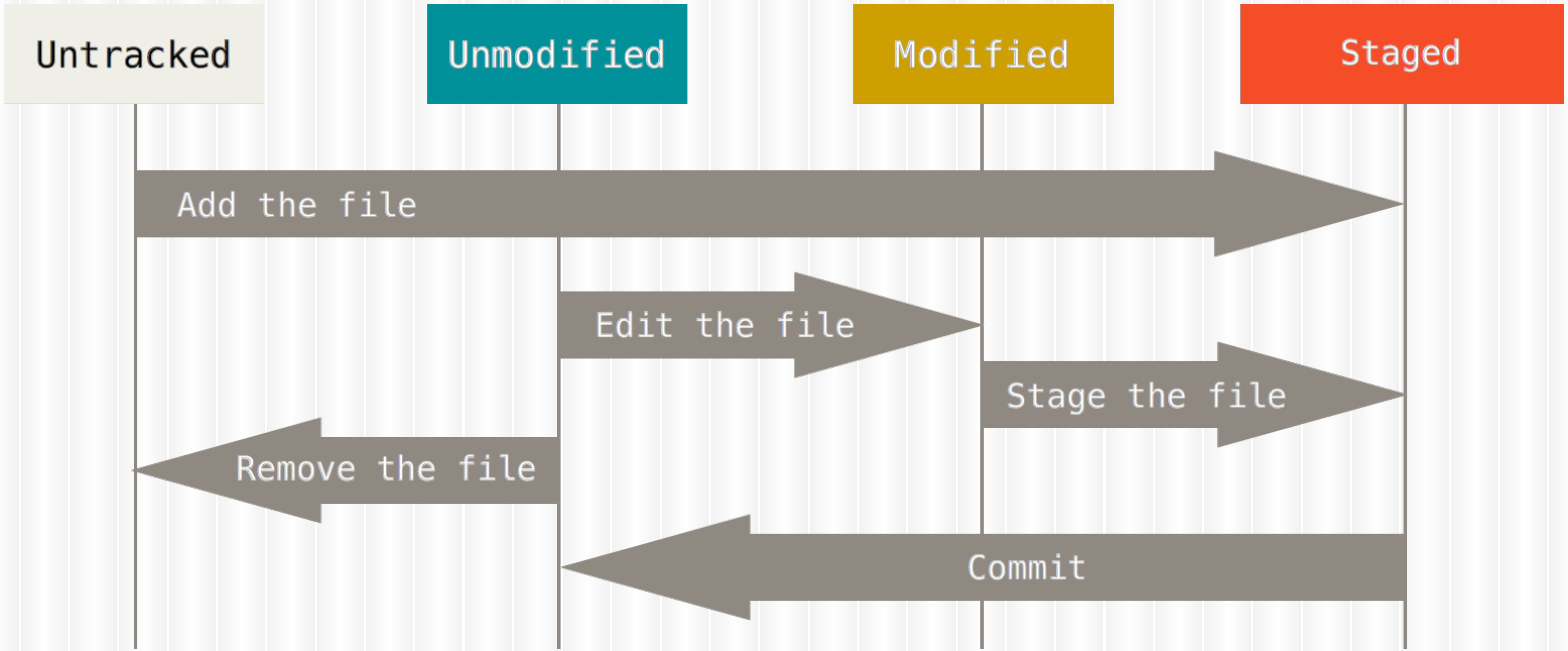



Checkins Over Time
Version 1
Version 2
Version 3
Version 4
Version 5
File A
File B
File C
A1
B
C1
A1
B
C2
A2
B1
C2
A2
B2
C3

Snapshots



Os três estágios



Estado dos arquivos

Instalação

Instalação - *nix

- Debian/Ubuntu

```
$ apt-get install git
```

- Fedora

```
$ yum install git
```

- Gentoo

```
$ emerge --ask --verbose dev-vcs/git
```

- Arch Linux

```
$ pacman -S git
```

- openSUSE

```
$ zypper install git
```

- FreeBSD

```
$ cd /usr/ports/devel/git
$ make install
```

- Solaris 11 Express

```
$ pkg install developer/versioning/git
```

- OpenBSD

```
$ pkg_add git
```

Instalação - Windows

- git-scm.com/download/win

Instalação - MAC

- git-scm.com/download/mac

Git no Windows

Git Bash é um terminal para Windows que recebe comandos de Linux

Pode ser usado para:
Criar diretórios (mkdir)
Navegar pelo sistema de arquivos (cd, ls)
Executar os comandos do git
(...)

É possível invocar os comandos do git diretamente nos diretórios, utilizando o menu do botão direito. Entretanto, um terminal do Git Bash será executado. Ou seja, funciona como um atalho.

Principais Comandos

Git Manual

```
$ git help
```

```
$ git help -a
```

```
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--
info-path]
           [-p|--paginate|--no-pager] [--no-replace-objects] [--
bare]
           [--git-dir=<path>] [--work-tree=<path>] [--
namespace=<name>]
           <command> [<args>]

The most commonly used git commands are:
   add        Add file contents to the index
   bisect     Find by binary search the change that introduced a bug
   branch     List, create, or delete branches
   checkout   Checkout a branch or paths to the working tree
   clone      Clone a repository into a new directory
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree,
etc
   fetch      Download objects and refs from another repository
   grep       Print lines matching a pattern
   init       Create an empty Git repository or reinitialize an
existing one
   log        Show commit logs
   merge      Join two or more development histories together
   mv         Move or rename a file, a directory, or a symlink
   pull       Fetch from and integrate with another repository or a
local branch
   push       Update remote refs along with associated objects
   rebase     Forward-port local commits to the updated upstream
head
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index
   show       Show various types of objects
   status     Show the working tree status
   tag        Create, list, delete or verify a tag object signed
with GPG

'git help -a' and 'git help -g' lists available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
```

Git Manual

```
$ git help <comando>
```

- Exemplo:

```
$ git help add
```

```
NAME
       git-add - Add file contents to the index

SYNOPSIS
       git add [-n] [-v] [--force | -f] [--interactive | -i] [--
patch | -p]
                 [--edit | -e] [--[no-]all | --[no-]ignore-removal |
[--update | -u]]
                 [--intent-to-add | -N] [--refresh] [--ignore-
errors] [--ignore-missing]
                 [--] [<pathspec>...]

DESCRIPTION
       This command updates the index using the current content
found in the working tree, to prepare the content staged for the
next commit. It
       typically adds the current content of existing paths as a
whole, but with some options it can also be used to add content with
only part of
       the changes made to the working tree files applied, or remove
paths that do not exist in the working tree anymore.

 (...)

OPTIONS
       <pathspec>...
           Files to add content from. Fileglobs (e.g.  *.c) can be
given to add all matching files. Also a leading directory name (e.g.
dir to add
           dir/file1 and dir/file2) can be given to add all files in
the directory, recursively.

       -n, --dry-run
           Don't actually add the file(s), just show if they exist
and/or will be ignored.

       -v, --verbose
           Be verbose.

       -f, --force
           Allow adding otherwise ignored files.
```

Definindo sua identidade

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

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

- Exemplo:

```
$ git config --global user.name “Brauner Oliveira”

$ git config --global user.email brauner@usp.br
```

null

Definindo editor de texto padrão

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

- Exemplo:

```
$ git config --global
core.editor gedit
```

null

Criando um repositório

- Crie um diretório com o nome: “repository”
- Vá até o diretório e execute o comando:

```
$ git init
```

```
Initialized empty Git repository in
/caminho/para/repository/.git/
```

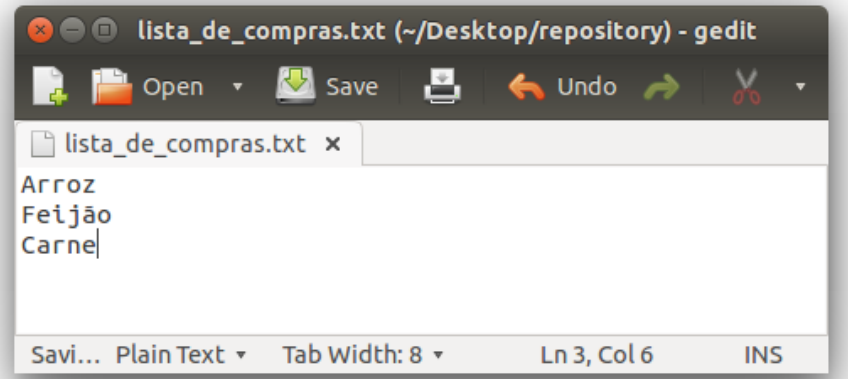
Criando um arquivo dentro do diretório

- Crie um arquivo dentro do diretório:

repository/lista_de_compras.txt

- Preencha-o com os seguintes itens:

  Arroz
  Feijão
  Carne



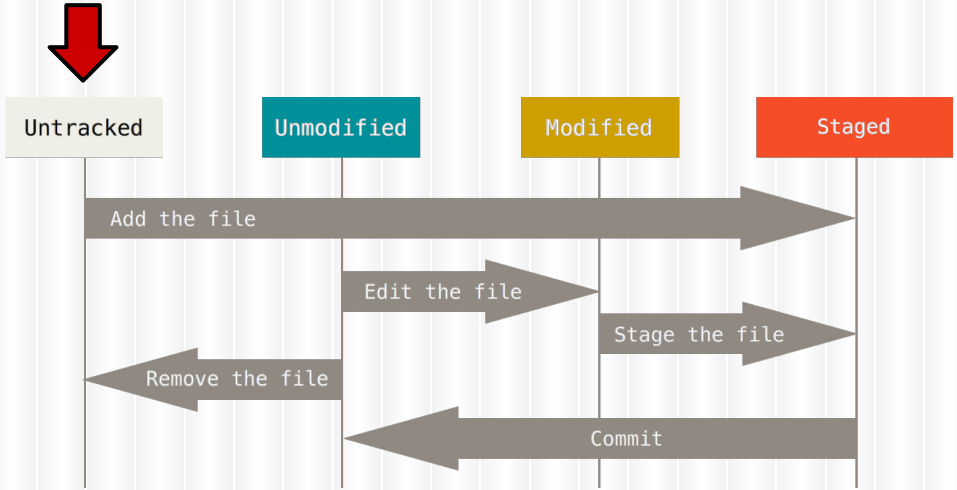
Verificando estado dos arquivos

```
$ git status
```

```
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be
committed)

        lista_de_compras.txt

nothing added to commit but untracked files present (use "git
add" to track)
```



Adicionando arquivos à staging area

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

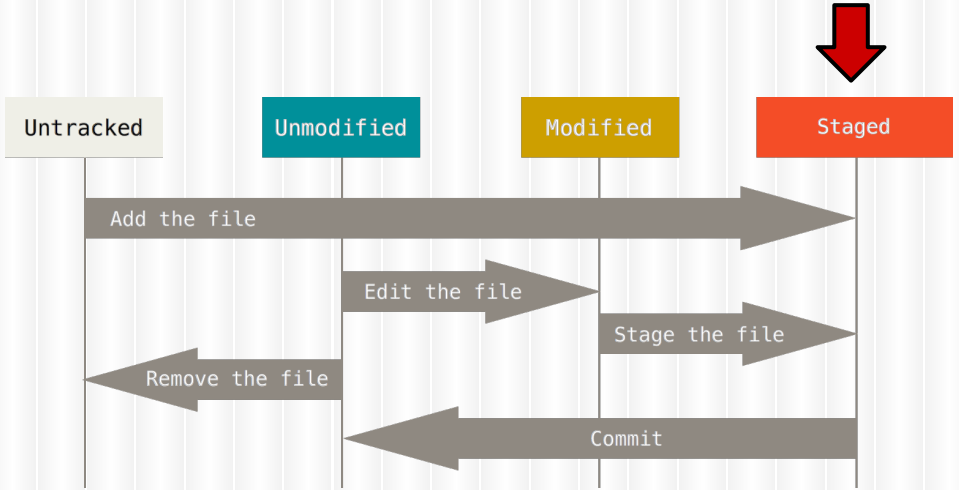
<arquivos> = {file_name, *, *.ext, /directory/*.ext}

- Exemplo:

```
$ git add lista_de_compras.txt
```

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   lista_de_compras.txt
```



Adicionando arquivos ao repositório

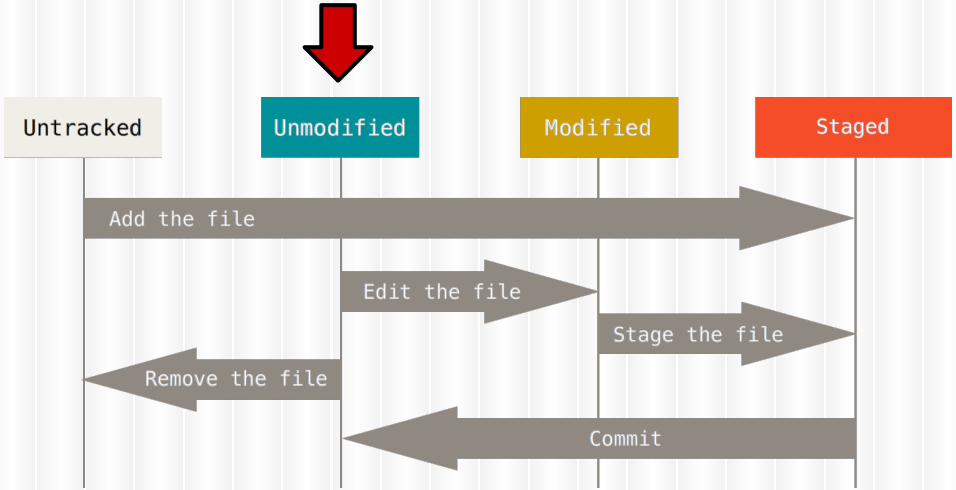
```
$ git commit

$ git commit -m
"<mensagem de commit>"

$ git commit -a -m
"<mensagem de commit>"

$ git commit -m
"Primeira lista de
compras"
```

```
[master ac76483] Primeira lista de compras
 1 file changed, 3 insertions(+)
 create mode 100644 lista_de_compras.txt
```



Editando arquivos

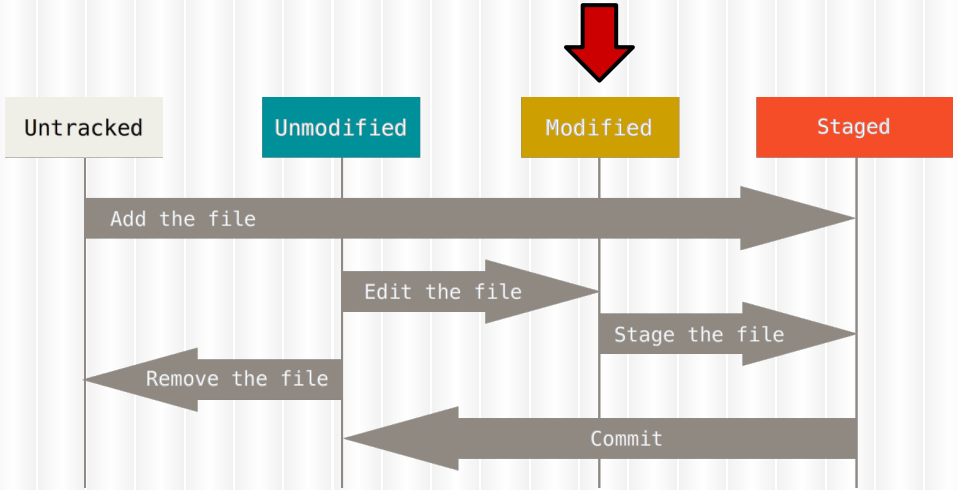
- Abra o arquivo lista_de_compras.txt
- Adicione “Batata” ao fim da lista e salve o arquivo

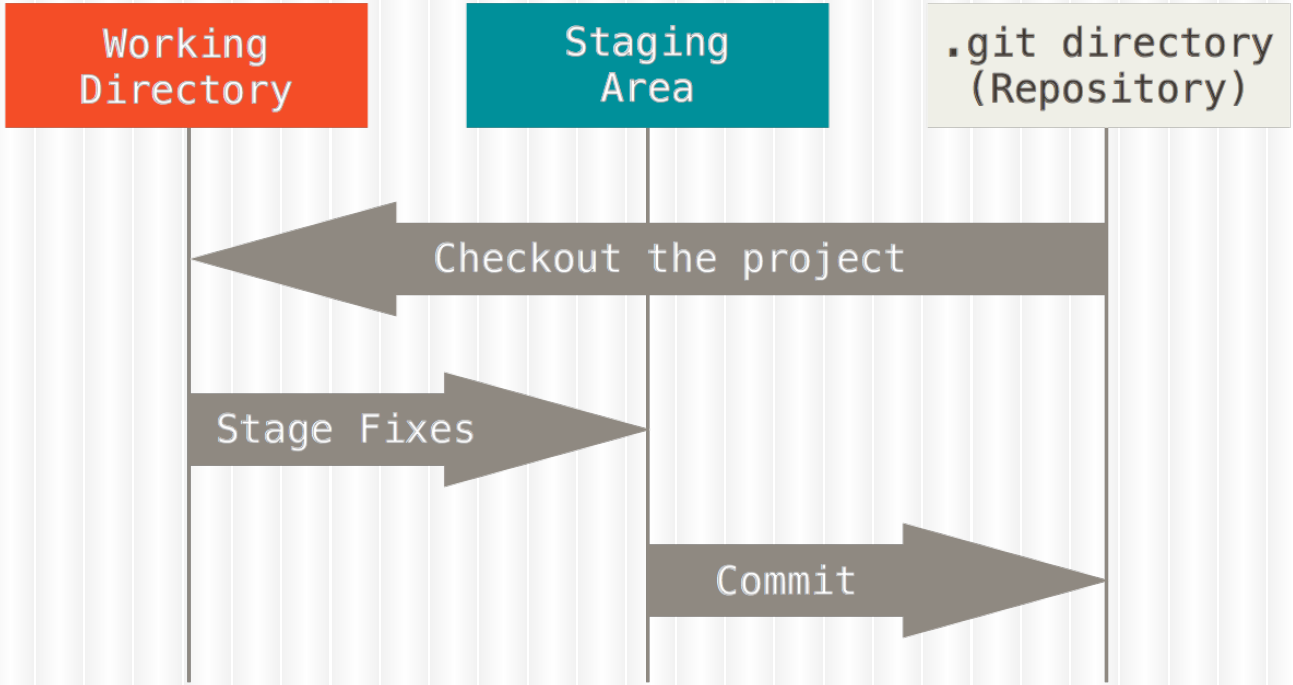
```
$ git status
```

```
$ git commit -a -m
“Batata adicionada à
lista”
```

```
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in
working directory)

        modified:   lista_de_compras.txt
```





Os três estágios - Por onde passamos?

Histórico de *commits*

```
$ git log

$ git log --oneline
```

```
commit f7d00ec4b2a08be7def79d8f68d2283a5bc9b9cb
Author: Brauner Oliveira <brauner.rno@gmail.com>
Date:   Thu May 7 08:57:12 2015 -0300

    Batata adicionada à lista

commit ac76483f7c674612e7d6e4cc87b3384940cd87a7
Author: Brauner Oliveira <brauner.rno@gmail.com>
Date:   Wed May 6 19:58:49 2015 -0300

    Primeira lista de compras

_____________________________________________


f7d00ec Batata adicionada à lista
ac76483 Primeira lista de compras
```

Diferenças de *commits*

```
$ git diff <commitA>
<commitB>
```

```
$ git diff ac76483 f7d00ec
```

```
$ git diff f7d00ec ac76483
```

```
diff --git a/lista_de_compras.txt
b/lista_de_compras.txt
index 30448a4..6917d3d 100644
--- a/lista_de_compras.txt
+++ b/lista_de_compras.txt
@@ -1,3 +1,4 @@
 Arroz
 Feijão
 Carne
+Batata
```

```
diff --git a/lista_de_compras.txt
b/lista_de_compras.txt
index 6917d3d..30448a4 100644
--- a/lista_de_compras.txt
+++ b/lista_de_compras.txt
@@ -1,4 +1,3 @@
 Arroz
 Feijão
 Carne
-Batata
```

Rotulando *commits*

```
$ git tag <rótulo>

$ git tag -a <rótulo>
-m <comentário>

$ git tag -a v1.0 -m
“Lista de compras para
o almoço”

$ git show v1.0
```

```
tag v1.0
Tagger: Brauner Oliveira <brauner.rno@gmail.com>
Date:   Thu May 7 09:28:40 2015 -0300

Lista de compras para o almoço

commit f7d00ec4b2a08be7def79d8f68d2283a5bc9b9cb
Author: Brauner Oliveira <brauner.rno@gmail.com>
Date:   Thu May 7 08:57:12 2015 -0300

    Batata adicionada à lista

diff --git a/lista_de_compras.txt
b/lista_de_compras.txt
index 30448a4..6917d3d 100644
--- a/lista_de_compras.txt
+++ b/lista_de_compras.txt
@@ -1,3 +1,4 @@
 Arroz
 Feijão
 Carne
+Batata
```

Git Branching

- Usado para criar novos fluxos de versionamento
- Comumente utilizado para separar artefatos em desenvolvimento dos que estão em produção

Gerenciamento de *branches*

- Listar *branches*

```
$ git branch
```

- Criar *branch*

```
$ git branch <branch_name>
```

```
$ git branch lista2
```

```
$ git branch -v
```

```
null
```

```
* master f7d00ec Batata adicionada à lista
  lista2 f7d00ec Batata adicionada à lista
```

Navegando pelos nós (*commits*)

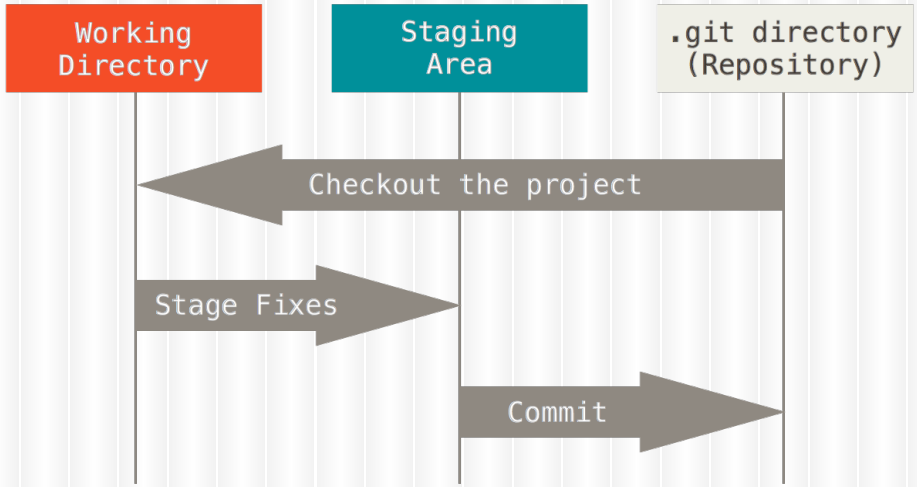
```
$ git checkout <branch_name>

$ git checkout <commit_hash>

$ git checkout <tag>

$ git checkout -b
<branch_name>

$ git checkout lista2
```

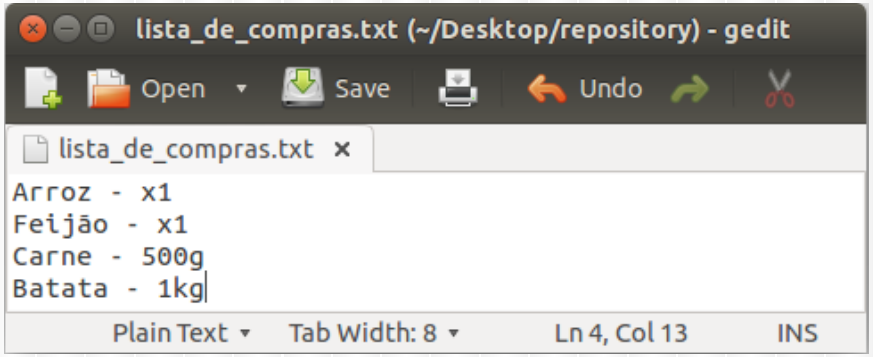


Modificando o arquivo no novo branch

- Edite o arquivo lista_de_compras.txt

- Adicione as quantidades que serão compradas para cada item:

Arroz - x1
Feijão - x1
Carne - 500g
Batata - 1kg



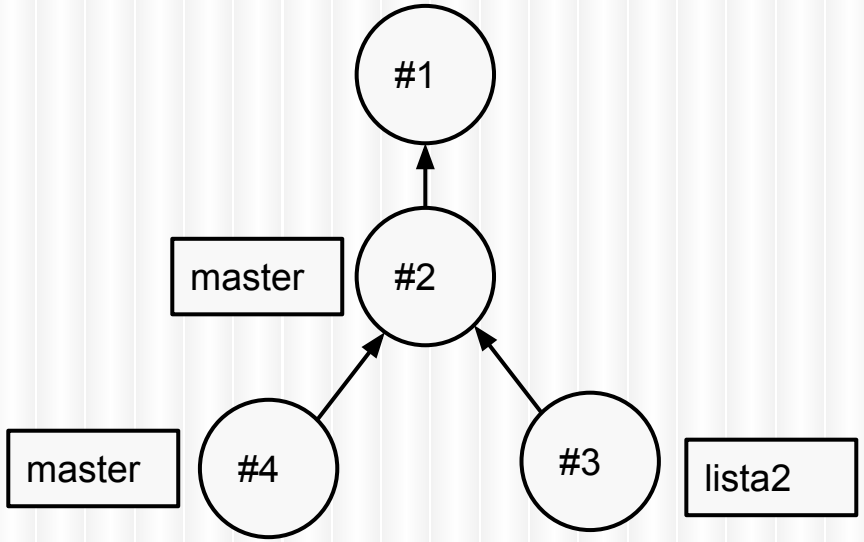
Enviando as mudanças para o repositório

```
$ git commit -a -m "Quantidades
para comprar adicionadas"
```

```
$ git checkout master
```

- Editar arquivo e adicionar à lista: Miojo e Lasanha, e remover Batata.

```
$ git commit -a -m "Adicionado:
miojo e lasanha. Removido:
batata"
```



Mesclando branches

```
$ git merge <branch_name>
```

```
$ git merge lista2
```

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

Resolvendo conflitos

master (HEAD)

```
Arroz
Feijão
Carne
Miojo
Lasanha
```

lista2

```
Arroz - x1
Feijão - x1
Carne - 500g
Batata - 1kg
```

lista_de_compras.txt (Com conflitos)

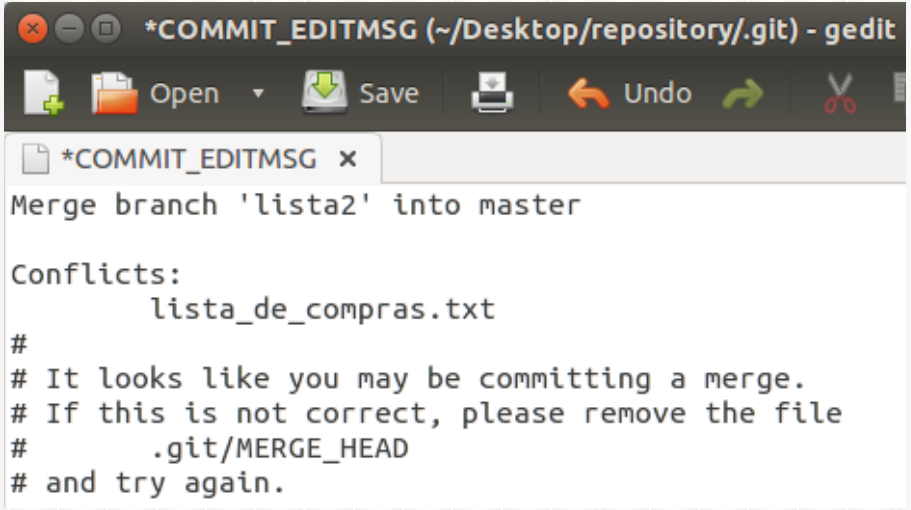
```
<<<<<<< HEAD
Arroz
Feijão
Carne
Miojo
Lasanha
=======
Arroz - x1
Feijão - x1
Carne - 500g
Batata - 1kg
>>>>>>> lista2
```

lista_de_compras.txt (Após resolver conflitos)

```
Arroz - x1
Feijão - x1
Carne - 500g
Miojo - 5x
Lasanha - 2x
```

Finalizando o merge

```
$ git commit -a
```




```
[master 050eaf0] Merge branch 'lista2'
```

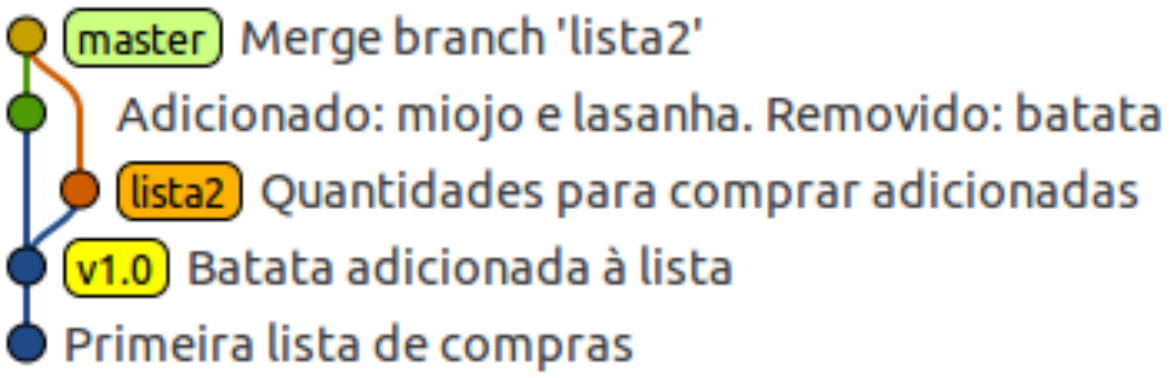
Verificando o histórico dos *branches*

```
$ git checkout master
$ git log --oneline

$ git checkout lista2
$ git log --oneline
```

```
9bc4bd8 Merge branch 'lista2'
6e685a9 Adicionado: miojo e lasanha. Removido: batata
1540598 Quantidades para comprar adicionadas
060af1d Batata adicionada à lista
a7b4adf Primeira lista de compras


1540598 Quantidades para comprar adicionadas
060af1d Batata adicionada à lista
a7b4adf Primeira lista de compras
```



Árvore final

Subject	Author	Date
master origin/HEAD origin/master Improve BRIN infra, minmax opclass and regression test	Alvaro Herrera	Qui 07 Mai 2015 13:02:22 BRT
Fix incorrect math in DetermineSafeOldestOffset.	Robert Haas	Qui 07 Mai 2015 12:00:47 BRT
origin/REL9_3_STABLE Fix incorrect math in DetermineSafeOldestOffset.	Robert Haas	Qui 07 Mai 2015 12:00:47 BRT
origin/REL9_4_STABLE Fix incorrect math in DetermineSafeOldestOffset.	Robert Haas	Qui 07 Mai 2015 12:00:47 BRT
Makefile: Add comment that doc uninstall clears man directories	Bruce Momjian	Qui 07 Mai 2015 11:26:08 BRT
origin/REL9_0_STABLE Properly send SCM status updates when shutting down service on Windows	Magnus Hagander	Qui 07 Mai 2015 10:04:13 BRT
origin/REL9_1_STABLE Properly send SCM status updates when shutting down service on Windows	Magnus Hagander	Qui 07 Mai 2015 10:04:13 BRT
origin/REL9_2_STABLE Properly send SCM status updates when shutting down service on Windows	Magnus Hagander	Qui 07 Mai 2015 10:04:13 BRT
Properly send SCM status updates when shutting down service on Windows	Magnus Hagander	Qui 07 Mai 2015 10:04:13 BRT
Properly send SCM status updates when shutting down service on Windows	Magnus Hagander	Qui 07 Mai 2015 10:04:13 BRT
Properly send SCM status updates when shutting down service on Windows	Magnus Hagander	Qui 07 Mai 2015 10:04:13 BRT
Fix indentation that could mask a future bug	Magnus Hagander	Qui 07 Mai 2015 06:41:26 BRT
Fix minor resource leak in pg_dump	Magnus Hagander	Qui 07 Mai 2015 06:40:15 BRT
Avoid using a C++ keyword as a structure member name.	Robert Haas	Ter 05 Mai 2015 23:41:03 BRT
citext's regexp_matches() functions weren't documented, either.	Tom Lane	Ter 05 Mai 2015 17:11:01 BRT
citext's regexp_matches() functions weren't documented, either.	Tom Lane	Ter 05 Mai 2015 17:11:01 BRT
citext's regexp_matches() functions weren't documented, either.	Tom Lane	Ter 05 Mai 2015 17:11:01 BRT
citext's regexp_matches() functions weren't documented, either.	Tom Lane	Ter 05 Mai 2015 17:11:01 BRT
citext's regexp_matches() functions weren't documented, either.	Tom Lane	Ter 05 Mai 2015 17:11:01 BRT
Backport "Add pgreadlink() on Windows to read junction points".	Robert Haas	Ter 05 Mai 2015 16:54:26 BRT
Fix incorrect declaration of citext's regexp_matches() functions.	Tom Lane	Ter 05 Mai 2015 16:50:53 BRT

Trecho do Repositório PostgreSQL

Clonando um repositório

```
$ git clone <url repositório>
```

- Fora de repository/, execute o comando:

```
$ git clone https://github.com/gabrielecirulli/2048.git
```

```
Cloning into '2048'...
remote: Counting objects: 1173, done.
remote: Total 1173 (delta 0), reused 0 (delta 0),
pack-reused 1173
Receiving objects: 100% (1173/1173), 591.37 KiB |
365.00 KiB/s, done.
Resolving deltas: 100% (660/660), done.
Checking connectivity... done.
```

Repositório remoto

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

- Exemplos de URL

ssh://brauner@10.62.45.80:22/caminho/para/repositorio

https://github.com/rails/rails.git

Repositório remoto

- Listar repositório remotos

```
$ git remote -v
```

- Vá até o repositório do jogo 2048 e execute o comando:

```
$ git remote -v
```

```
null
```

```
origin https://github.com/gabrielecirulli/2048.git (fetch)
origin https://github.com/gabrielecirulli/2048.git (push)
```

Atualizando o repositório local

```
$ git remote add github https://github.
com/
brauneroliveira/repository.git
```

```
$ git fetch <remote>
```

```
$ git fetch github
```

```
$ git checkout master
$ git merge github/master
```

```
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 2 (delta 0),
pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.
com/brauneroliveira/repository
 * [new branch]      master     -> github/master
```

Atualizando o repositório local (2)

```
$ git pull <remote> <branch>
```

```
$ git pull github master
```

```
From https://github.com/brauneroliveira/repository
 * branch            master     -> FETCH_HEAD
Updating 9bc4bd8..0ebe1a1
Fast-forward
 lista_de_compras.txt | 1 +
 1 file changed, 1 insertion(+)
```

Atualizando o repositório remoto

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

```
$ git push github master
```

- Se push for negado, use git fetch ou git pull para incorporar as mudanças que foram feitas por outra pessoa

```
Username for 'https://github.com': brauneroliveira
Password for 'https://brauneroliveira@github.com':
Counting objects: 18, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (9/9), done.
Writing objects: 100% (18/18), 1.59 KiB | 0 bytes/s, done.
Total 18 (delta 2), reused 0 (delta 0)
To https://github.com/brauneroliveira/repository.git
 * [new branch]      master -> master
```

Git GUI

- EGit - Plugin para eclipse
- gitg - GNOME GUI

- Outras GUIs: git-scm.com/downloads/guis

GitHub

Recursos

- Servidor de repositórios (público | privado)
- Fork
- Pull request
- Issue tracking
- Wiki
- Diff interface (plain text, images)
- Social coding (contribution history, post-like communication)

GitHub

Quem está no GitHub?

- https://github.com/torvalds/linux
- https://github.com/Microsoft
- https://github.com/facebook
- https://github.com/postgres/postgres
- https://github.com/twbs/bootstrap
- Etc…

Alternativa para o GitHub: https://bitbucket.org
(Repositório privado gratuito para até 5 usuários)

Material complementar

Try Git

try.github.io

Pro Git (2nd Edition)

http://git-scm.com/book/en/v2/

Udacity: How to Use Git and Github

udacity.com/course/how-to-use-git-and-github--ud775