

UT 2.2 GIT. Comandos básicos

ENTORNOS DE DESARROLLO

1. Modo de trabajo

- Existen varias maneras de poder trabajar con GIT

```
Símbolo del sistema
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.

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

These are common Git commands used in various situations:

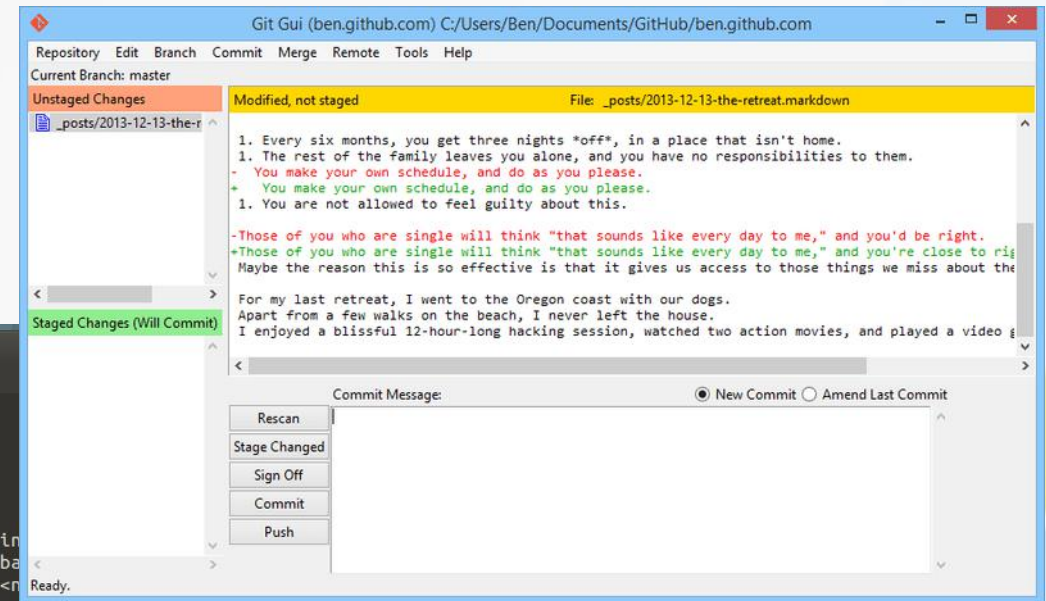

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree

examine the history and state (see also: git help log)
  bisect     Use binary search to find the first bad commit
  diff       Show changes between commits, commit and working tree, etc.
```

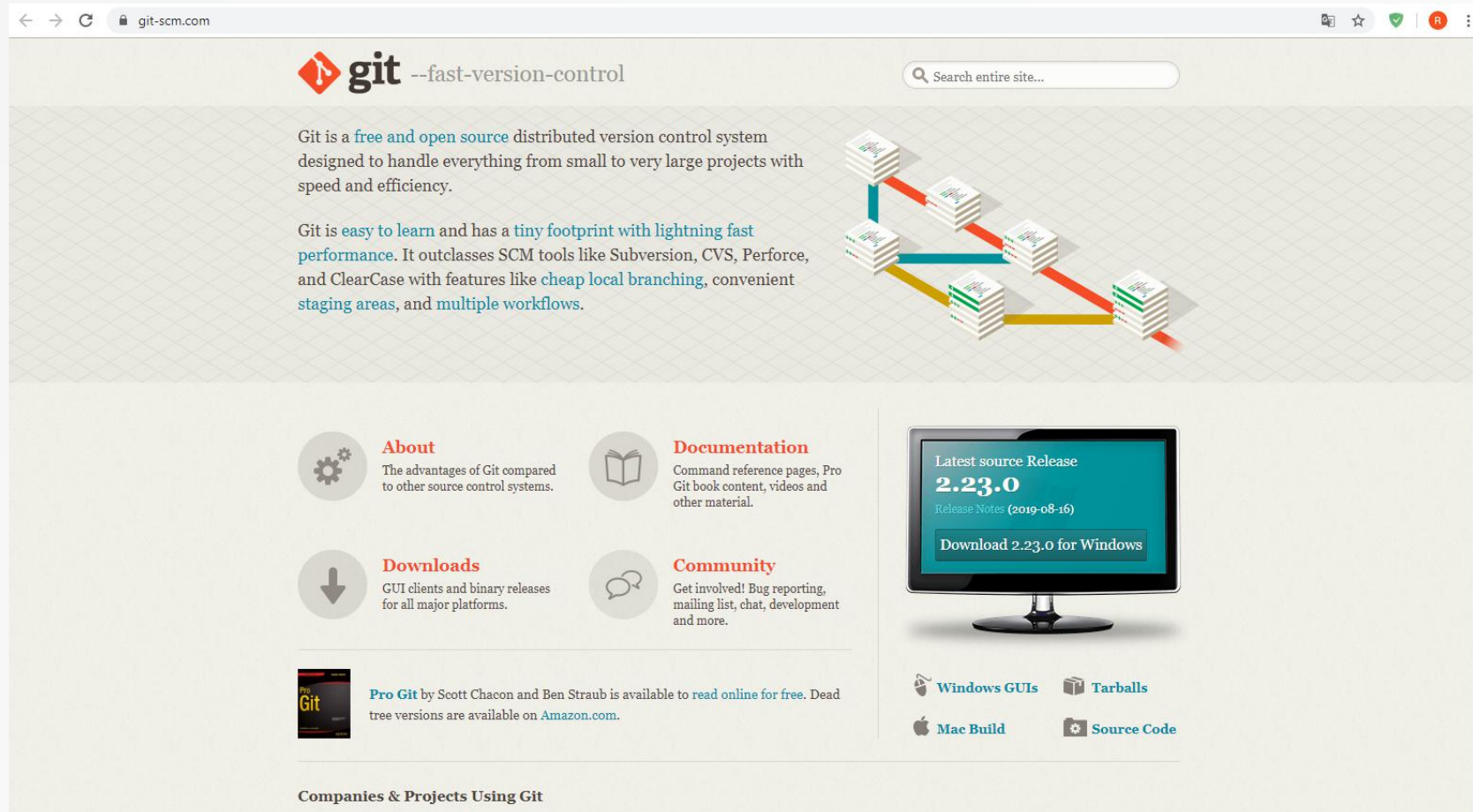
```
administrador@Ubuntu:~$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
       [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
       [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
       [--git-dir=<path>] [--work-tree=<path>] [--namespace=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

Els ordres de git més freqüentment utilitzats són:
add      Afegeix els continguts dels fitxers a l'índex
bisect   Troba per cerca binària el canvi que hagi introduït un defecte
branch   Llista, crea o suprimeix branques
checkout Agafa una rama o unes rutes a l'arbre de treball
clone    Clona un dipòsit a un directori nou
commit   Registra els canvis al dipòsit
diff     Mostra els canvis entre comissions, la comissió i l'arbre de treball, etc.
fetch    Baixa objectes i referències d'un altre dipòsit
grep     Imprimeix les línies coincidents amb un patró
init     Crea un dipòsit de Git buit o reinicialitza un existent
log      Mostra els registres de comissió
```



1. Modo de trabajo

- Web oficial: <https://git-scm.com>

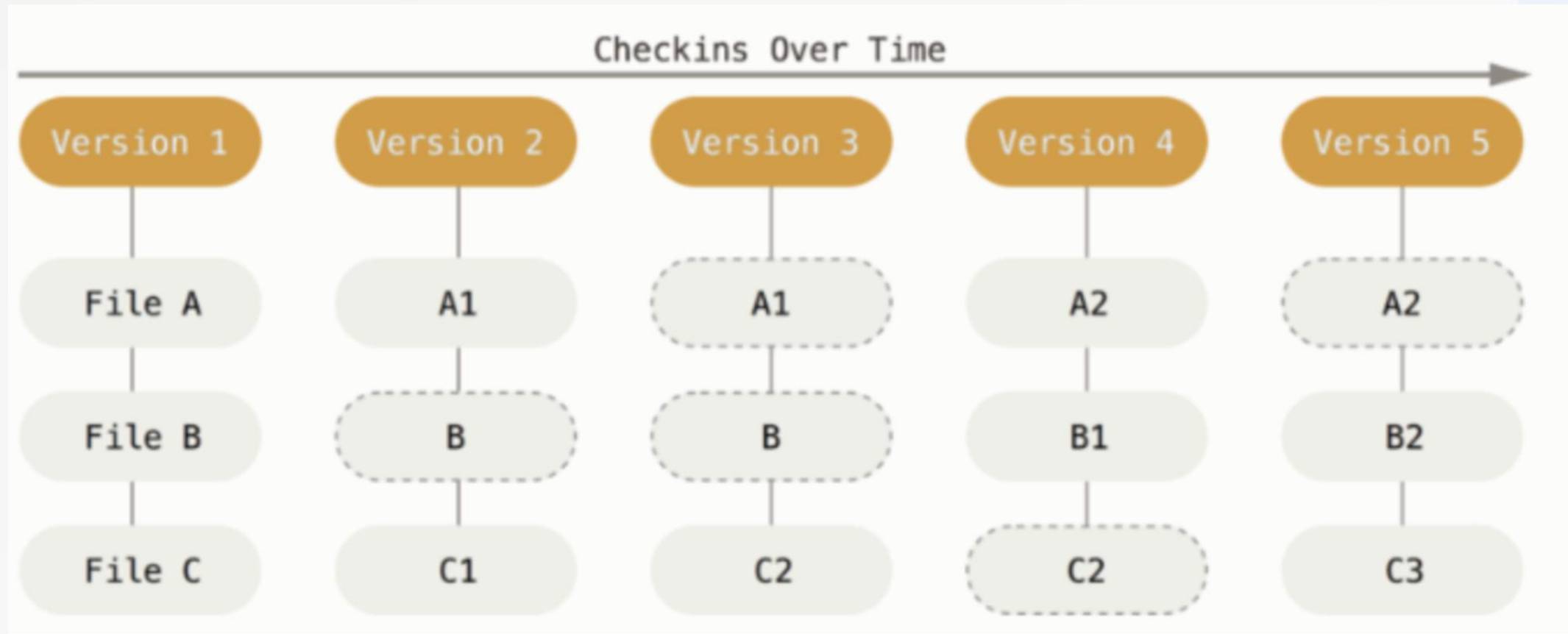


2. Fundamentos

2.1 Copias instantáneas

- Almacenamiento de copias instantáneas no habiendo diferencias en los archivos
- Para aumentar eficiencia si el archivo no se ha modificado creará un enlace al archivo anterior idéntico

2. Fundamentos



2. Fundamentos

2.2 Operaciones en almacenamiento local

- Toda la historia del proyecto se almacena en tu disco local por lo que las operaciones son inmediatas.
- Sin retardos de red.
- Puedes consultar la historia y realizar cambios y confirmarlos sin necesidad de estar conectado a la red.



2. Fundamentos

2.3 Integridad

- Las acciones son verificadas e identificadas mediante un checksum por lo que es imposible cambiar el contenido de los archivos o directorios sin que git lo sepa.
- El checksum se calcula mediante un hash SHA1

2. Fundamentos

2.4 Recuperación de la información

- Las acciones se basan en añadir información a la base de datos.
- Es muy difícil que el sistema realice una acción que no pueda revertirse.

3. Trabajando con GIT

- Instrucción de ayuda genérica

```
$ git help
```

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

...

3. Trabajando con GIT

- Instrucción de ayuda específica

```
$ git help config
```

```
GIT-CONFIG(1)
```

```
Git Manual
```

```
GIT-CONFIG(1)
```

Válido para cualquier comando de git

NAME

git-config - Get and set repository or global options

SYNOPSIS

```
git config [<file-option>] [type] [-z|--null] name [value [value_regex]]
git config [<file-option>] [type] --add name value
```

...

3. Trabajando con GIT

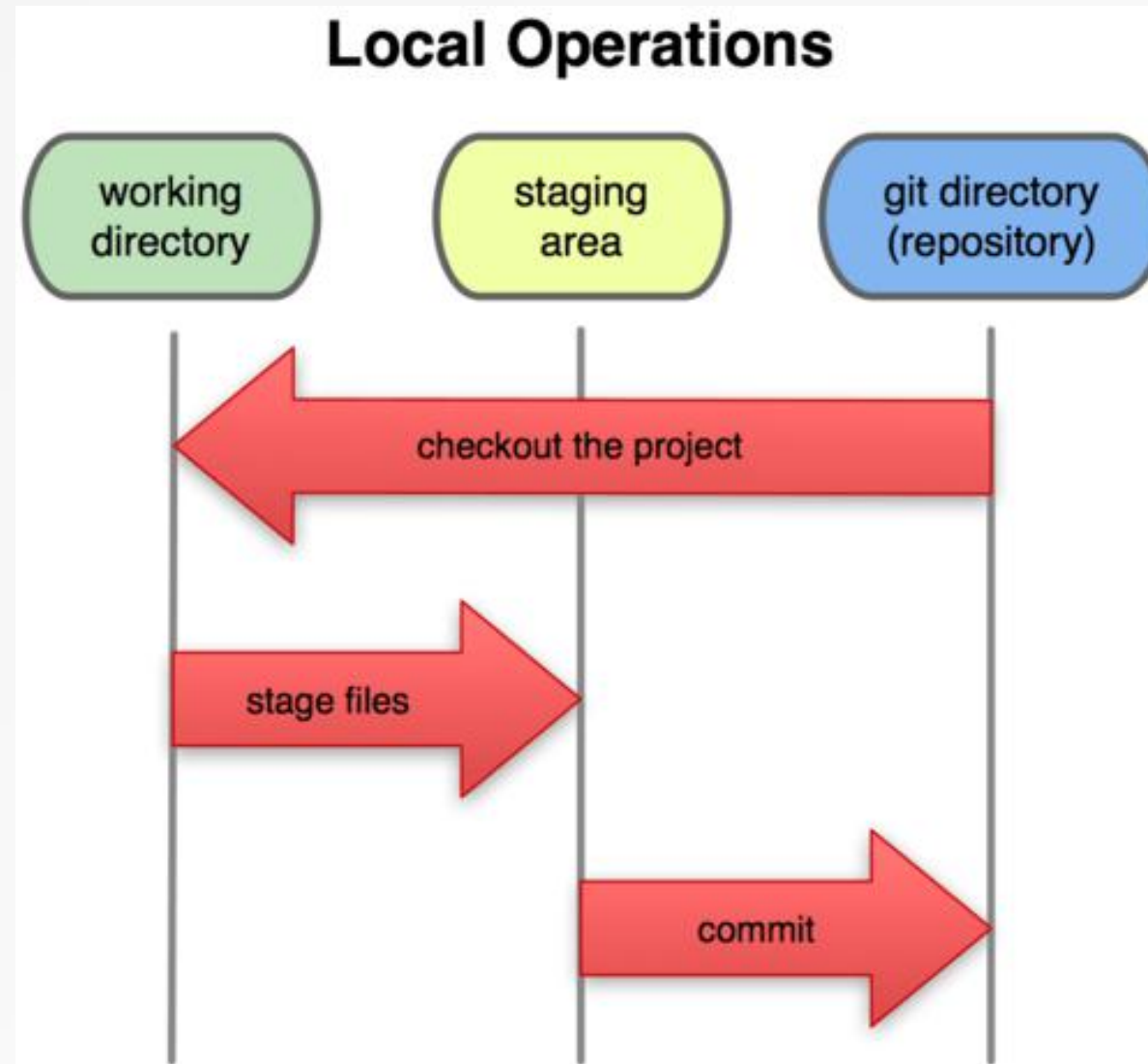


3. Estado de los archivos

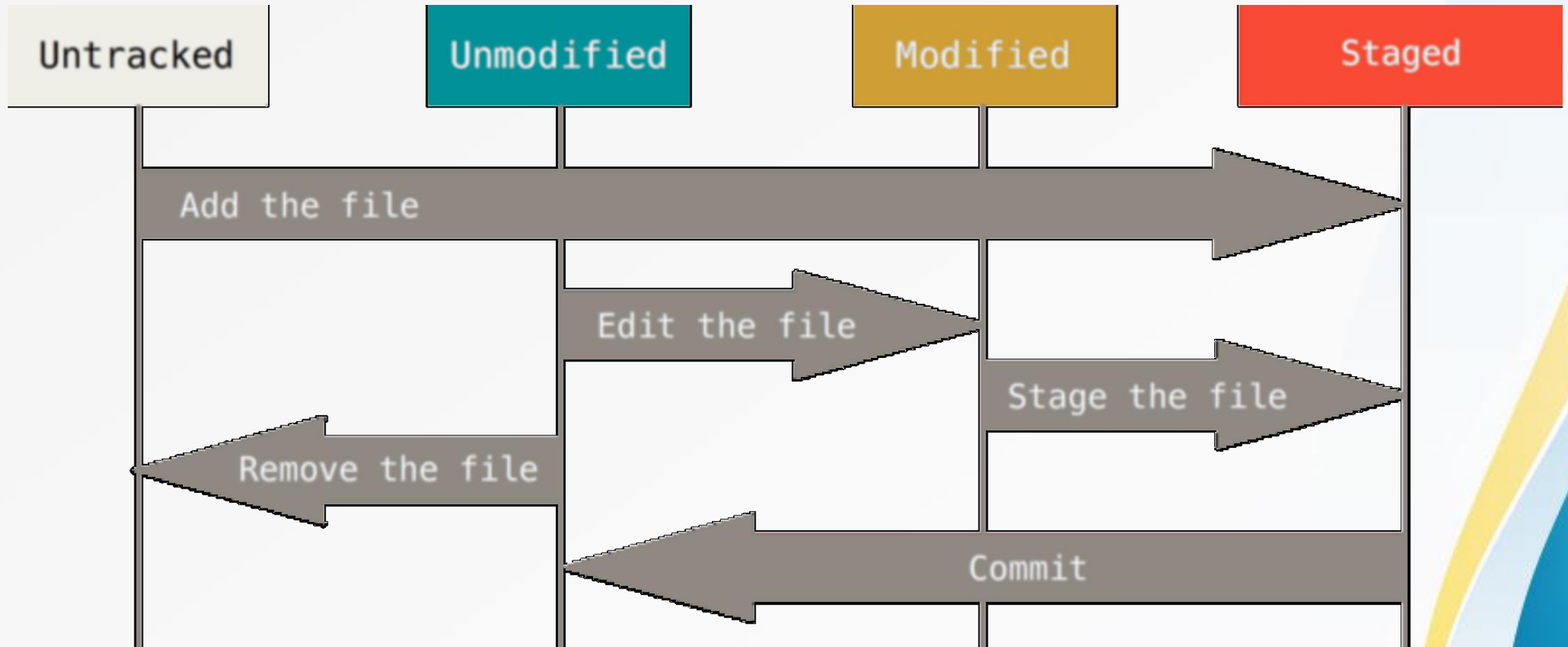
En GIT, hay tres estados en los que pueden encontrarse tus archivos sometidos a control de versión:

- **Confirmado (committed):** Los datos están almacenados de manera segura en la Bd.
- **Modificado (modified):** El archivo está modificado pero no confirmado en la Bd.
- **Preparado (staged):** Marcado para que vaya en la próxima confirmación

3. Estado de los archivos



3. Estado de los archivos



4. Configuración inicial

Añadir los datos de usuario

```
$ git config --global user.name "Gregg Pollack"
```

El autor de los cambios realizados

```
$ git config --global user.email gregg@codeschool.com
```

El email del autor de los cambios

```
$ git config --global color.ui true
```

Algunos textos en color

4. Configuración inicial

Comienzo de un repositorio

```
$ mkdir store
```

```
$ cd store
```

```
$ git init
```

```
Initialized empty Git repository in /Users/gregg/store/.git/
```

Los metadatos de Git son almacenados aquí



5. Flujo de trabajo básico

- El usuario “Pepe” crea un fichero README.txt
 - El fichero comienza como sin seguimiento (untracked)
- Añadir el fichero a la zona de ficheros preparados
 - Prepararlo para que se le tome una “foto” al fichero
- Confirmar los cambios
 - “Sacar la foto” a todo lo que esté preparado



5. Flujo de trabajo básico

- Pepe modifica el fichero README.txt y añade uno nuevo llamado LICENSE
- Añade ambos archivos a la zona de preparados
- Se confirman ambas novedades



5. Flujo de trabajo básico (comandos)

- El usuario “Pepe” crea un fichero README.txt

```
$ git status ← Para chequear qué ha cambiado desde el último 'commit'
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#  README.txt ← El archivo que ha creado Pepe
nothing added to commit but untracked files present (use "git add" to track)
```


5. Flujo de trabajo básico (comandos)

- Añadir el fichero a la zona de ficheros preparados

```
$ git add README.txt
```

```
$ git status
```

```
# On branch master
```

```
#
```

```
# Initial commit
```

```
#
```

```
# Changes to be committed:
```

```
#   (use "git rm --cached <file>..." to unstage)
```

```
#
```

```
#   new file:   README.txt
```

```
#
```



El fichero ya está preparado

5. Flujo de trabajo básico (comandos)

- Confirmar los cambios

Mensaje de confirmación
¿Qué trabajo se ha hecho?

```
$ git commit -m "Create a README."
```

```
[master abe28da] Create a README.  
1 files changed, 1 insertions(+), 0 deletions(-)  
create mode 100644 README.txt
```



Línea del tiempo

1

```
$ git status  
# On branch master  
nothing to commit (working directory clean)
```

master

No hay ficheros nuevos o modificados desde el último commit

5. Flujo de trabajo básico (comandos)

- Pepe modifica el fichero README.txt y añade uno nuevo llamado LICENSE

```
$ git status
# On branch master
# Changed but not updated:
#
#   modified:   README.txt
#
# Untracked files:
#
#   LICENSE
no changes added to commit
```

Master

1




5. Flujo de trabajo básico (comandos)

- Añade ambos archivos a la zona de preparados

```
$ git add README.txt LICENSE
```

O bien

```
$ git add --all
```



Añade todos los ficheros nuevos o modificados

```
$ git status

# On branch master
# Changes to be committed:
#
#   new file:   LICENSE
#   modified:   README.txt
#
```


5. Flujo de trabajo básico (comandos)

- Se confirman ambas novedades

```
$ git commit -m "Add LICENSE and finish README."
```

```
[master 1b0019c] Add LICENSE and finish README.  
2 files changed, 21 insertions(+), 0 deletions(-)  
create mode 100644 LICENSE
```



Master

2

1



6. Histórico de GIT

```
$ git log
```

```
commit 1b0019c37e3f3724fb2e9035e6bab4d7d87bf455
```

```
Author: Gregg Pollack <gregg@codeschool.com>
```

```
Date: Thu Jul 5 22:31:27 2012 -0400
```

```
Add LICENSE and finish README.
```

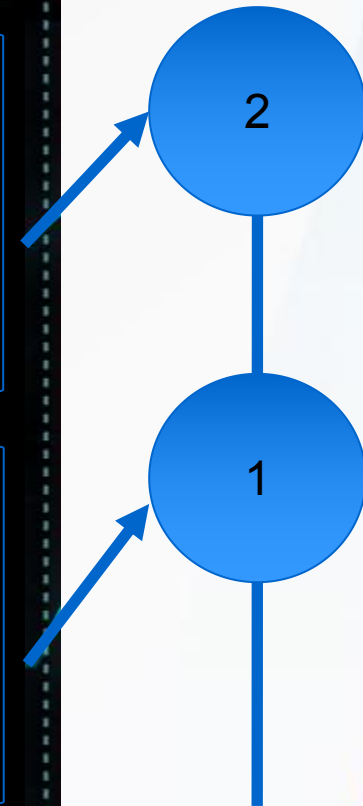
```
commit 5acaf86b04aaf9cbbb8ebb9042a20a46d0b9ce76
```

```
Author: Gregg Pollack <gregg@codeschool.com>
```

```
Date: Thu Jul 5 22:00:46 2012 -0400
```

```
Create a README.
```

Master



7. Otros modos de añadir

```
$ git add <list of files>
```

```
$ git add --all
```

```
$ git add *.txt
```

```
$ git add docs/*.txt
```

```
$ git add docs/
```

```
$ git add "*.txt"
```