Workshop, Git

v1.0, Alejandro M. Bernardis

figment

+ Introduction
+ Source Code Management

AGENDA

INTRODUCTION

SOURCE CODE MANAGEMENT

Git

git-scm.com

git --distributed-even-if-your-workflow-isnt

Search entire site...

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

Learn Git in your browser for free with Try Git.

About

The advantages of Git compared to other source control systems.

Documentation

Command reference pages, Pro Git book content, videos and other material.

Downloads

GUI clients and binary releases for all major platforms.

Community

Get involved! Mailing list, chat, development and more.

Mac GUIs Tarballs

Pro Git by Scott Chacon is available to read online for free. Dead tree versions are available on Amazon.com.

http://git-scm.com/

Code School - Try Git

try.github.io/levels/1/challenges/1

1.1 · Got 15 minutes and want to learn Git?

tryGit

Git allows groups of people to work on the same documents (often code) at the same time, and without stepping on each other's toes. It's a distributed version control system.

Our terminal prompt below is currently in an **octobox** directory. To initialize a Git repository here, type the following command:

```
git init
```

TryGit—1160x300

```
Press enter to submit commands
$
```

My Octobox Repository

Advice

http://try.github.io/levels/1/challenges/1

Git - Instalando Git

git-scm.com/book/es/Empezando-Instalando-Git

git --distributed-is-the-new-centralized

Search entire site...

About
Documentation
- Reference
- Book
- Blog
- Videos
- External Links

Blog
Downloads
Community

Download this book in PDF, mobi, or ePub form for free.

This book is translated into German, Chinese, French, Japanese, Dutch, Russian, Korean, Brazilian Portugese and Czech.

Partial translations available in Arabic, Spanish, Indonesian, Italian, Macedonian, Polish, Turkish, Taiwanese Mandarin, Spanish (Nicaragua), Catalan, Thai and Finnish.

Chapters

1.4 Empezando - Instalando Git

Instalando Git

Vamos a empezar a usar un poco de Git. Lo primero es lo primero: tienes que instalarlo. Puedes obtenerlo de varias maneras; las dos principales son instalarlo desde código fuente, o instalar un paquete existente para tu plataforma.

Instalando desde código fuente

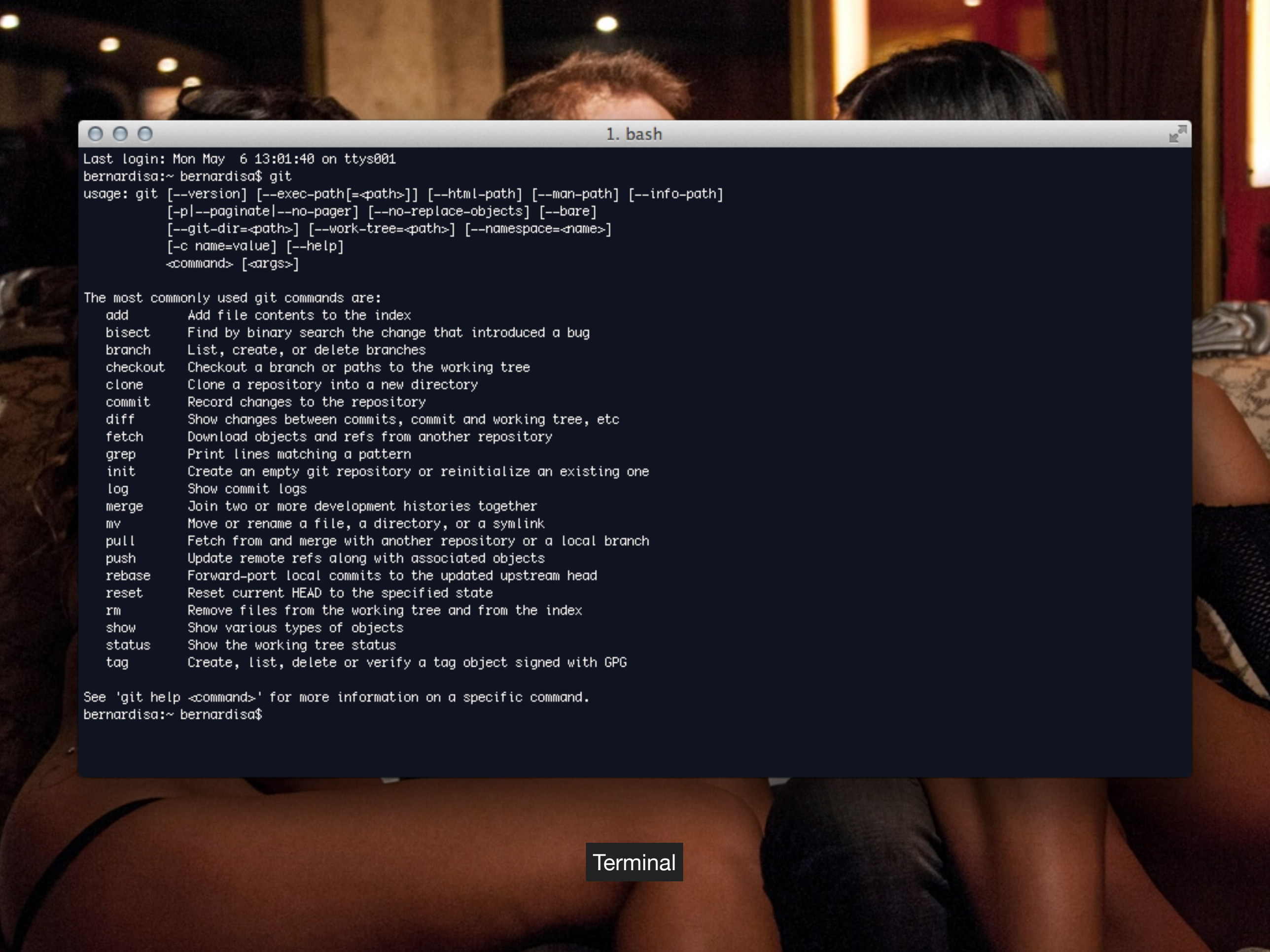
Si puedes, en general es útil instalar Git desde código fuente, porque obtendrás la versión más reciente. Cada versión de Git tiende a incluir útiles mejoras en la interfaz de usuario, por lo que utilizar la última versión es a menudo el camino más adecuado si te sientes cómodo compilando software desde código fuente. También ocurre que muchas distribuciones de Linux contienen paquetes muy antiguos; así que a menos que estés en una distribución muy actualizada o estés usando backports, instalar desde código fuente puede ser la mejor apuesta.

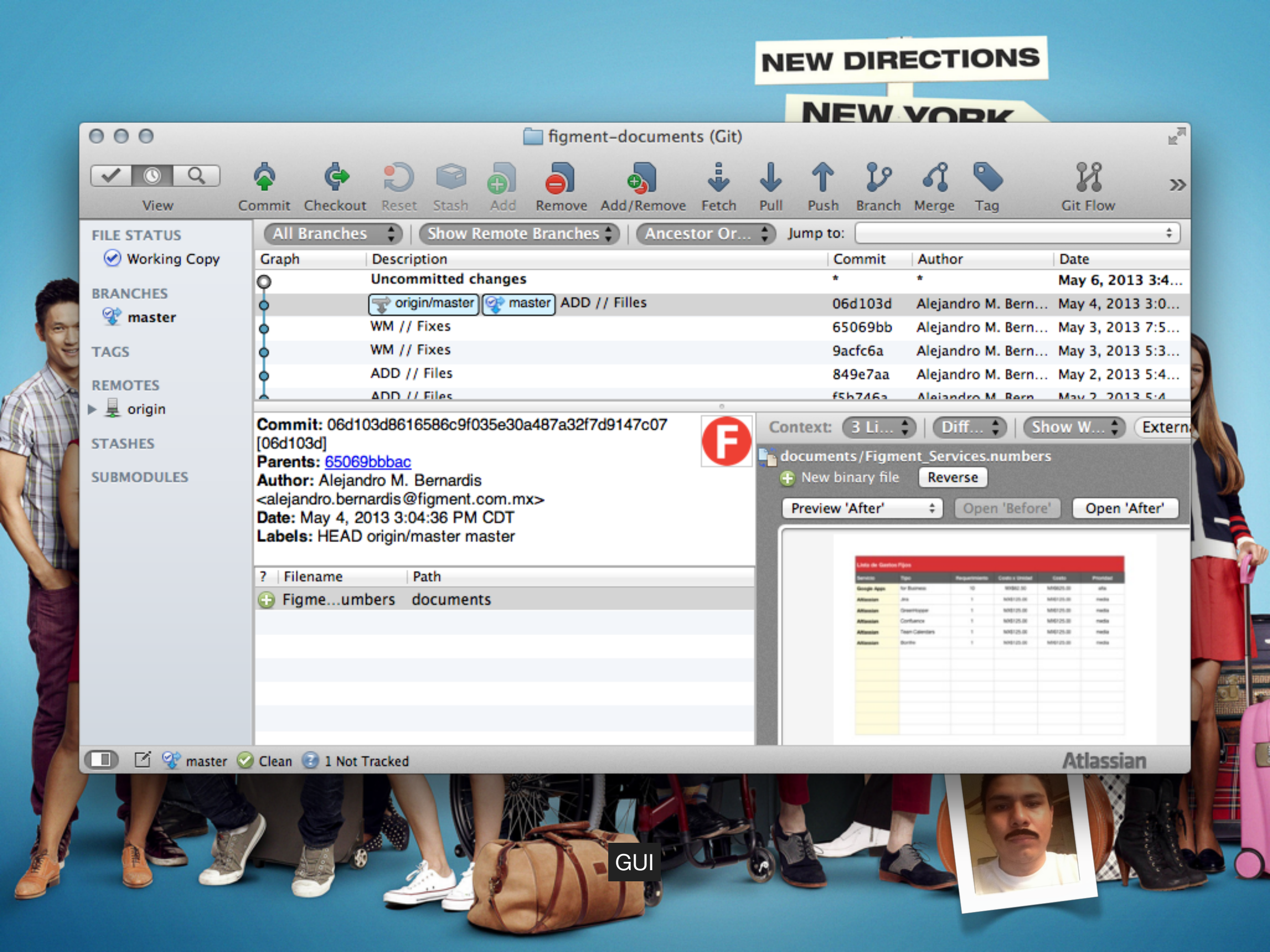
Para instalar Git, necesitas tener las siguientes librerías de las que Git depende: curl, zlib, openssl, expat, y libiconv. Por ejemplo, si estás en un sistema que tiene yum (como Fedora) o apt-get (como un sistema basado en Debian), puedes usar uno de estos comandos para instalar todas las dependencias:

```
$ yum install curl-devel expat-devel gettext-devel \
  openssl-devel zlib-devel

$ apt-get install libcurl4-gnutls-dev libexpat1-dev gettext \
  libz-dev
```

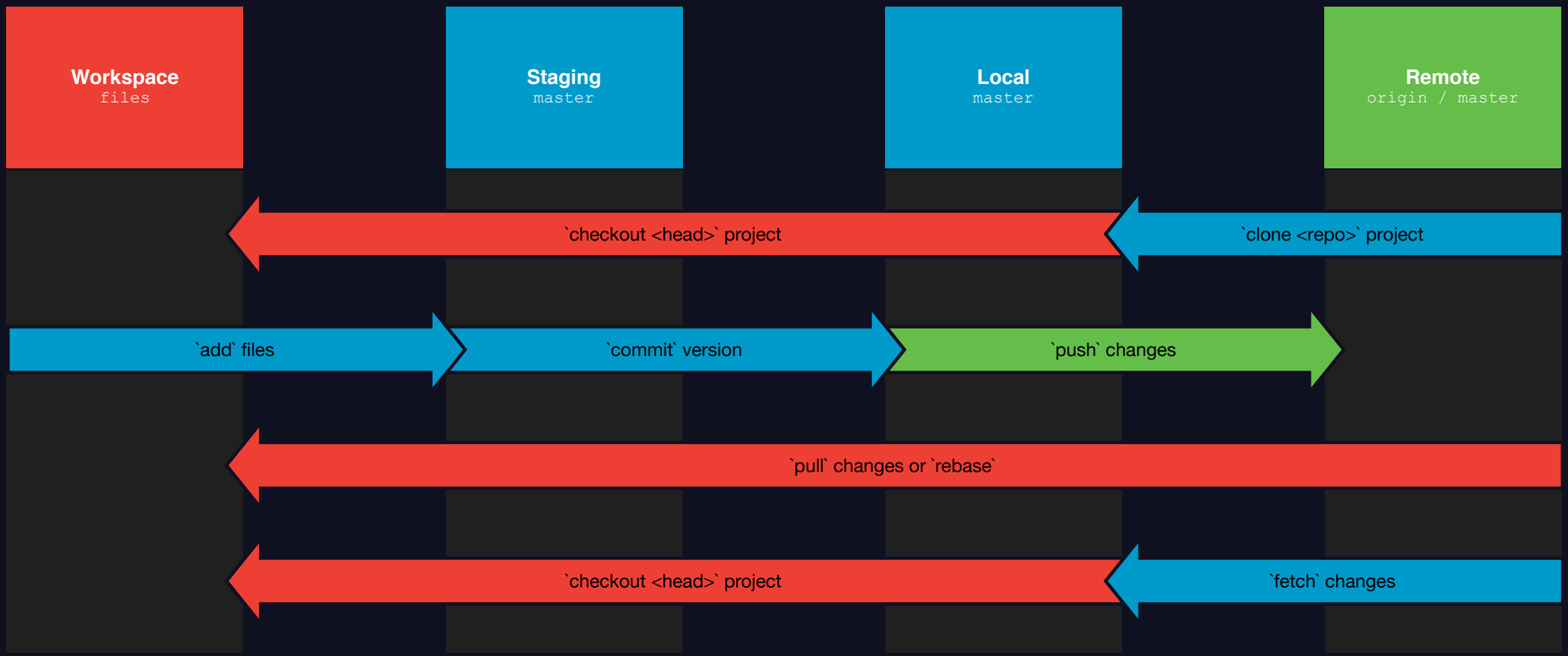
Cuando tengas todas las dependencias necesarias, puedes descargar la versión más reciente de Git

http://git-scm.com/book/es/Empezando-Instalando-Git


1. bash
Last login: Mon May 6 13:01:40 on ttys001
bernardisa:~ bernardisa$ git
usage: git [--version] [--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>]
           [-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
   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 merge 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
See 'git help <command>' for more information on a specific command.
bernardisa:~ bernardisa$
Terminal


NEW DIRECTIONS
NEW YORK
figment-documents (Git)
View
Commit
Checkout
Reset
Stash
Add
Remove
Add/Remove
Fetch
Pull
Push
Branch
Merge
Tag
Git Flow
FILE STATUS
Working Copy
BRANCHES
master
TAGS
REMOTES
origin
STASHES
SUBMODULES
All Branches
Show Remote Branches
Ancestor Or...
Jump to:
Graph
Description
Commit
Author
Date
Uncommitted changes
origin/master
master
ADD // Filles
06d103d
Alejandro M. Bern...
May 4, 2013 3:0...
WM // Fixes
65069bb
May 3, 2013 7:5...
WM // Fixes
9acfc6a
May 3, 2013 5:3...
ADD // Files
849e7aa
May 2, 2013 5:4...
Commit: 06d103d8616586c9f035e30a487a32f7d9147c07
[06d103d]
Parents: 65069bbbac
Author: Alejandro M. Bernardis
<alejandro.bernardis@figment.com.mx>
Date: May 4, 2013 3:04:36 PM CDT
Labels: HEAD origin/master master
Context:
3 Li...
Diff...
Show W...
Extern
documents/Figment_Services.numbers
New binary file
Reverse
Preview 'After'
Open 'Before'
Open 'After'
?
Filename
Path
Figme...umbers
documents
master
Clean
1 Not Tracked
Atlassian
GUI

FIRST STEPS



http://git-scm.com/book/es/Fundamentos-de-Git-Consejos-y-trucos

```
$ cd ~
$ ls -ld .* | grep git
-rw-r--r--  user  group  May 8 10:00  .gitconfig
-rw-r--r--  user  group  May 8 10:00  .gitignore_global
```

```
$ vim ~/.gitconfig
[user]
        name = Alejandro M. Bernardis
        email = alejandro.bernardis@figment.com.mx
[core]
        excludesfile = /home/bernardisa/.gitignore_global
        editor = mate -w
    ~
    ~
    ~
    ~
```

```
$ vim ~/.gitignore_global
# Examples
# --------
*~
*.ext
.file
.folder
folder
file.ext


# OS generated files
# ------------------
.DS_Store
.DS_Store?
._*
.Spotlight-V100
.Trashes
Icon?
ehthumbs.db
Thumbs.db
~
~
~
~
```

https://github.com/github/gitignore https://help.github.com/articles/ignoring-files

COMMANDS

```
# Configuración
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

http://git-scm.com/book/es/Empezando-Configurando-Git-por-primera-vez

```
# Creación
$ mkdir -p ~/projects/git-my-first-repo
$ cd ~/projects/git-my-first-repo

# Inicialización
$ git init

# Crear archivos: `.gitignore` and `README.md`
$ touch .gitignore README.md

# Agregar y Commitear
$ git add .
$ git commit -m ‘GIT // Ignore and Readme Files’
```

http://git-scm.com/book/es/Fundamentos-de-Git-Obteniendo-un-repositorio-Git

```
# Creación
$ mkdir -p ~/projects
$ cd ~/projects

# Inicialización
$ git clone git@domain.com:user/git-my-first-repo.git

# Crear archivos: `.gitignore` and `README.md`
$ touch .gitignore README.md

# Agregar y Commitear
$ git add .
$ git commit -m 'GIT // Ignore and Readme Files'
```

http://git-scm.com/book/es/Fundamentos-de-Git-Obteniendo-un-repositorio-Git

```
# Repositorio remoto: <repo>
git@bitbucket.org:figment_mexico/figment-workshop-git.git
```

```
# Creación
$ mkdir -p ~/projects
$ cd ~/projects

# Inicialización
$ git clone <repo>

# Crear archivos: `.gitignore` and `README.md`
$ touch .gitignore README.md

# Agregar y Commitear
$ git add .
$ git commit -m ‘GIT // Ignore and Readme Files’
```

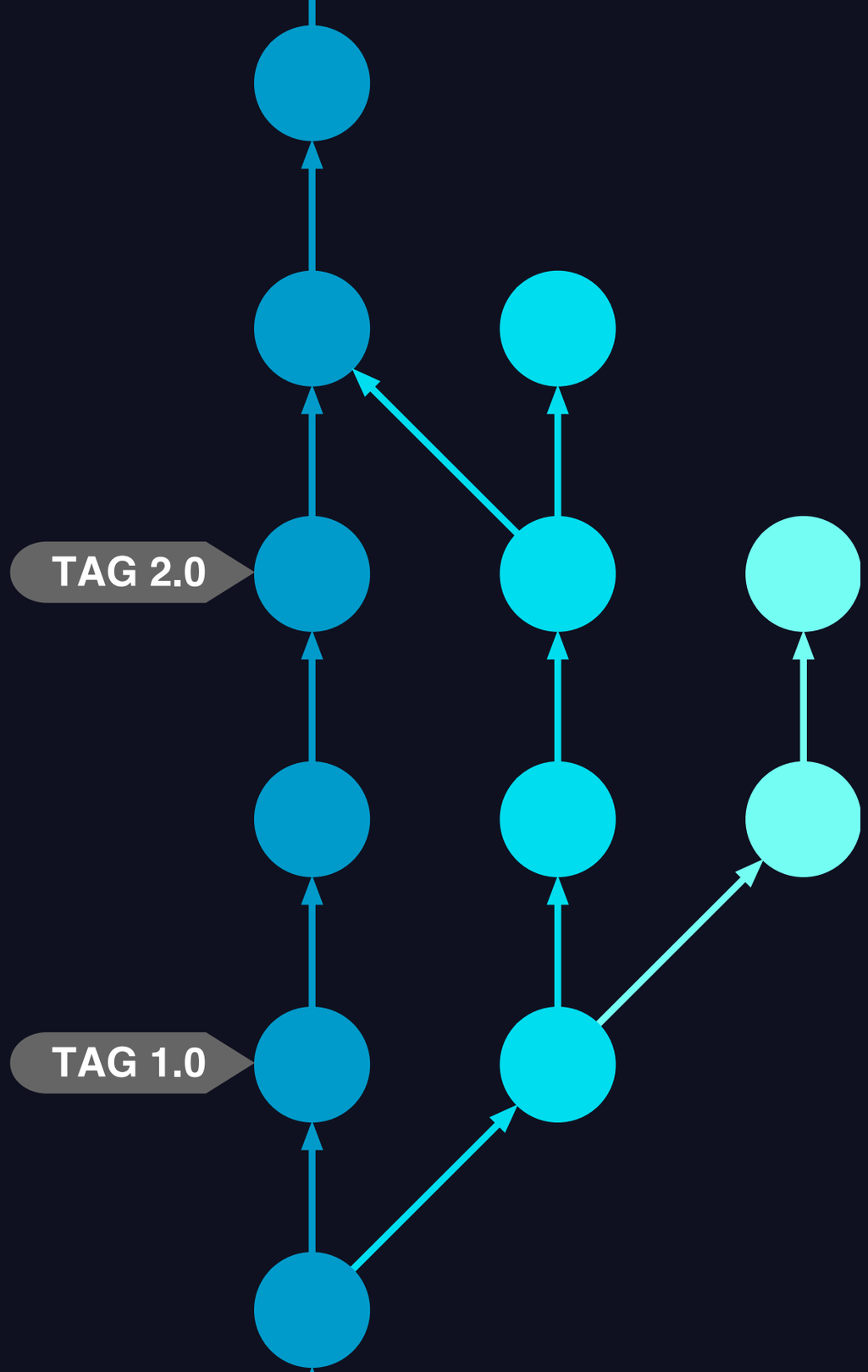
```
# Visualizar estado
$ git status
~
# On branch master
nothing to commit (working directory clean)

# Crear archivo: `AUTHORS.md`
$ touch AUTHORS.md

# Visualizar estado
$ git status
~
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   AUTHORS.md
nothing added to commit but untracked files present (use "git add" to track)
```

```
# Agregar archivo: `AUTHORS.md`
$ git add AUTHORS.md
$ git commit -m 'GIT // Authors File'

# Visualizar estado
$ git status
~
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   new file:   AUTHORS.md
```

```
# Visualizar registros
$ git log
~
commit 3b0151323a7a2263e57b928b3e37712ea5290959
Author: Alejandro M. Bernardis <alejandro.bernardis@figment.com.mx>
Date:   Mon May 6 17:06:31 2013 -0500

    GIT // Ignore and Readme Files
```

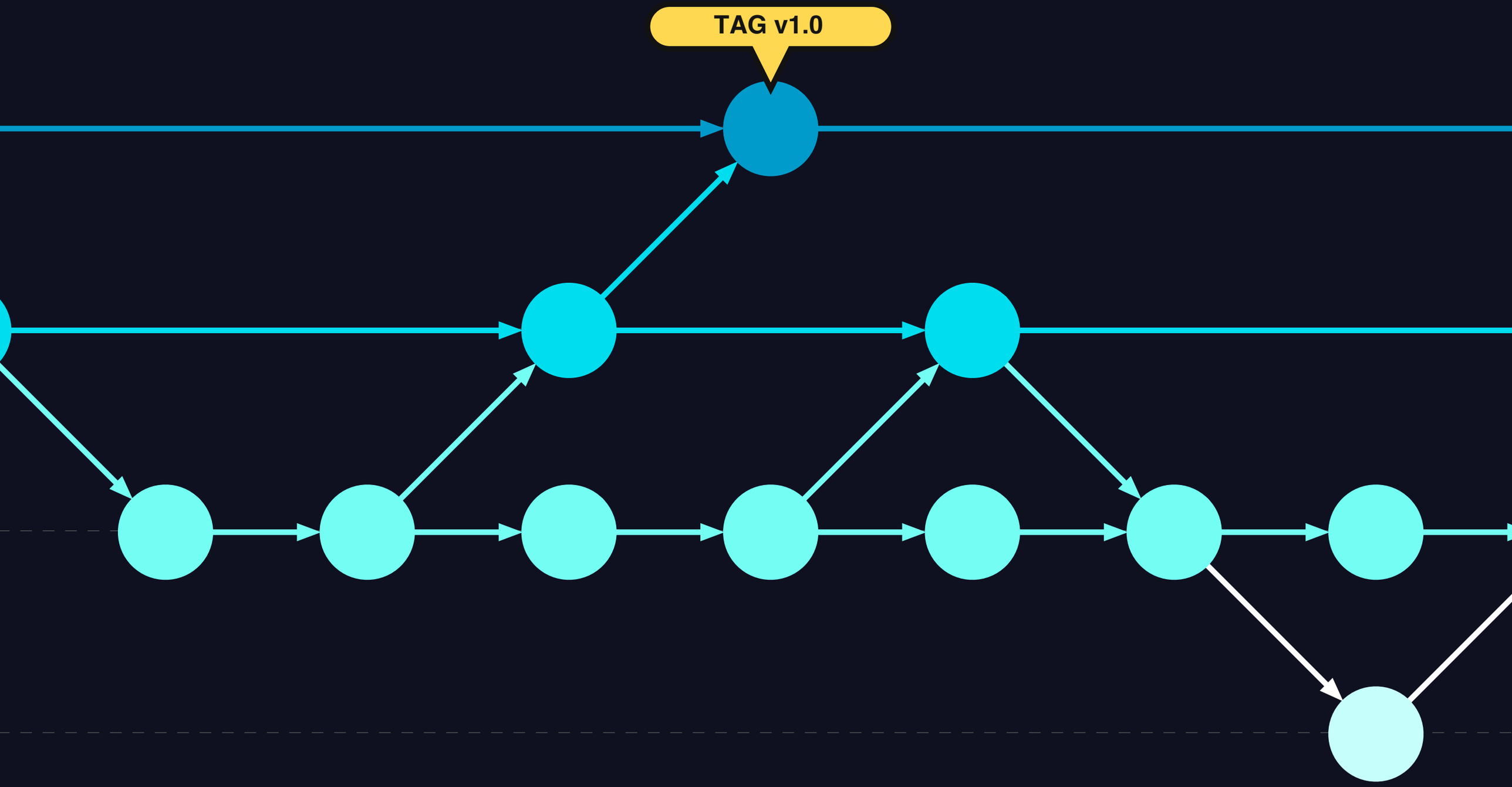
```
# Envío al repositorio remoto
$ git push origin master
~
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 260 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@bitbucket.org:figment_mexico/figment-workshop-git.git
 * [new branch]      master -> master
```

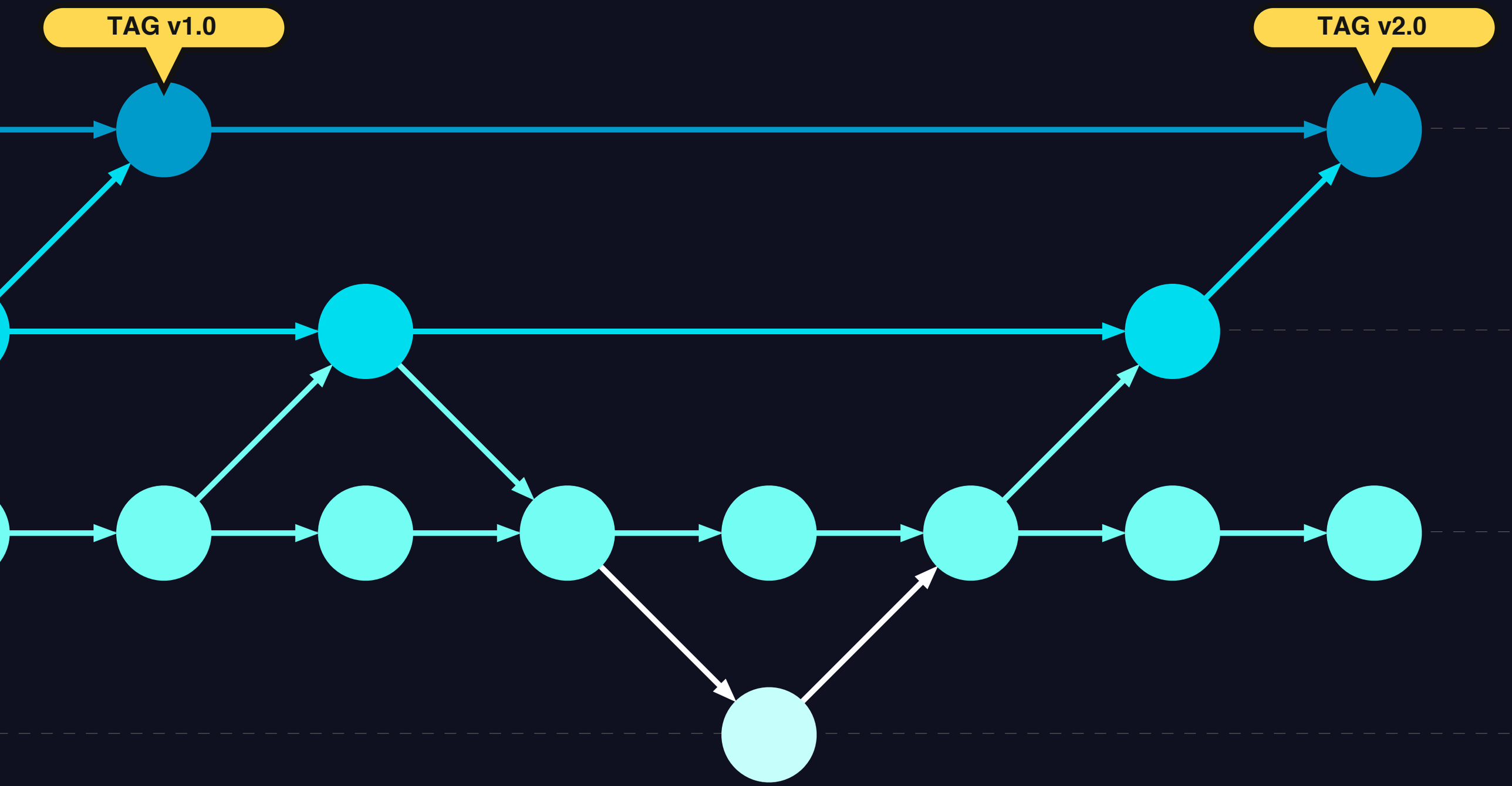
```
# Recuperar actualizaciones
$ git pull <origin>
~
remote: Counting objects: 39, done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 22 (delta 8), reused 0 (delta 0)
Unpacking objects: 100% (22/22), done.
From bitbucket.org:figment_mexico/figment-workshop-git
   e4442ec..9c3df21  master     -> origin/master
```

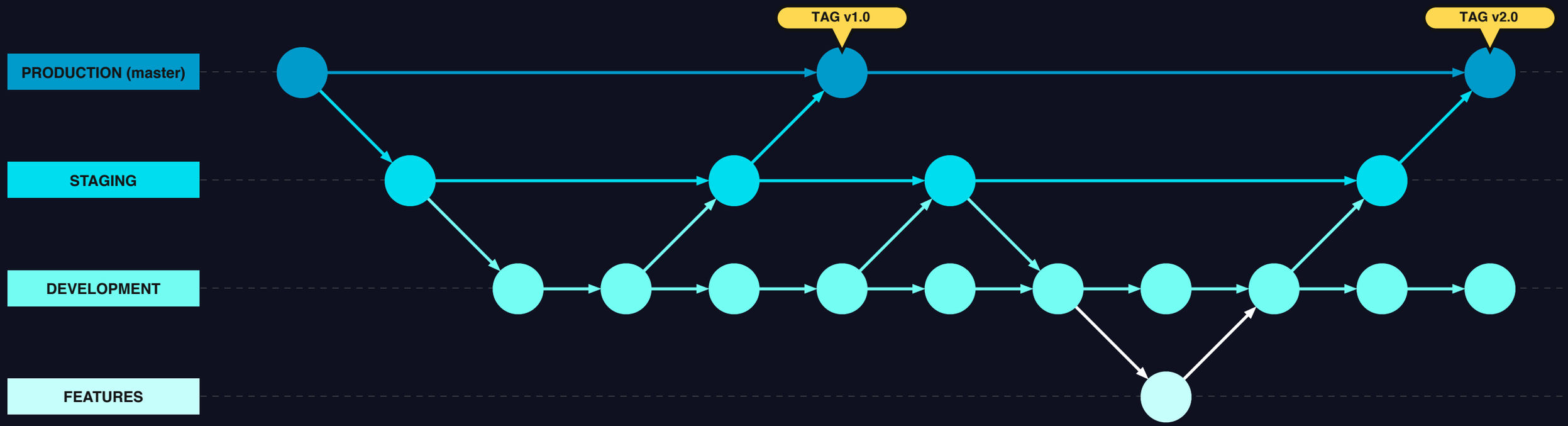


BRANCHS & TAGS


PRODUCTION (master)
STAGING
DEVELOPMENT
FEATURES


TAG v1.0


TAG v1.0
TAG v2.0


TAG v1.0
TAG v2.0
PRODUCTION (master)
STAGING
DEVELOPMENT
FEATURES

```
# Verificar rama
$ git branch -v

# Crear rama
$ git branch <new_branch>
$ git checkout -b <new_branch>

# Cambiar rama
$ git checkout <existing_branch>

# Unir ramas
$ git checkout <branch>
$ git merge <branch>

# Eliminar ramas
$ git branch -d <branch>

# Compartir ramas
$ git push origin <branch>

# Actualizar ramas
$ git pull origin <branch>
```

BRANCH

http://git-scm.com/book/es/Ramificaciones-en-Git

```
# Recuperar datos remotos
$ git fetch origin

# Trackear rama remota
$ git checkout --track origin/<branch>

# Crear rama desde otra remota
$ git checkout -b <new_branch> origin/<remote_branch>
$ git checkout -b <some_name> origin/<remote_branch>

# Eliminar rama remota
$ git push origin :serverfix
```

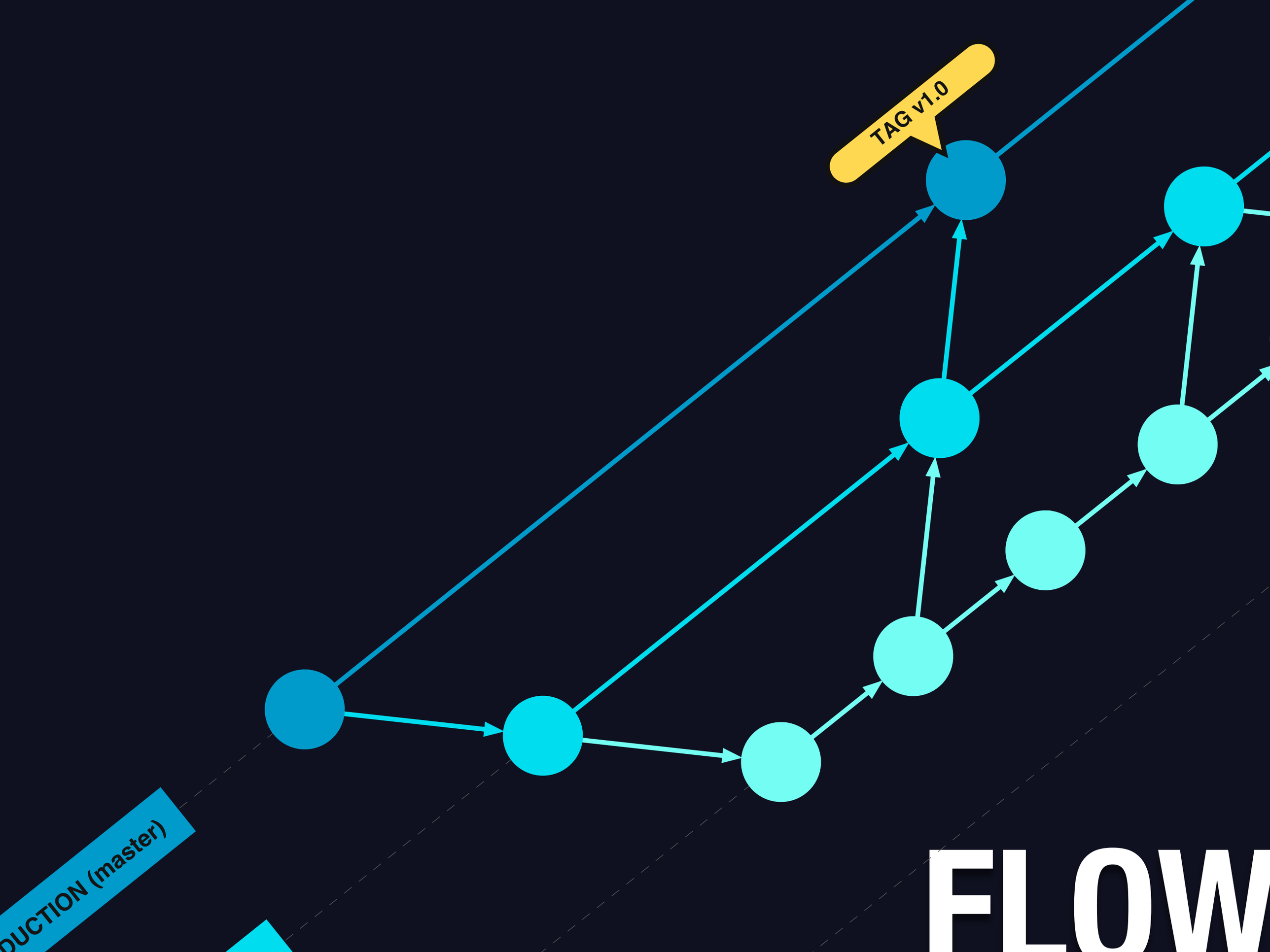
BRANCH

http://git-scm.com/book/es/Ramificaciones-en-Git

TAG

```
# Crear tag
$ git tag <value>

# Crear tag anotado
$ git tag -a <value> -m '<message>'

# Buscar tag
$ git tag -l <value>

# Visualizar tag
$ git show <value>

# Cambiar tag
$ git checkout <value>

# Compartir tag
$ git push origin <value>

Ej#: <value> = v1.0
```

http://git-scm.com/book/es/Fundamentos-de-Git-Creando-etiquetas


TAG v1.0
PRODUCTION (master)

LOCAL

DEVELOPMENT

STAGING

PRODUCTION

Creación del Reposistorio

Definición de Ramas

Clone <STA>

Clone <DEV>

Clone <DEV>

Commit <DEV>

Push <origin> <DEV>

Pull <origin> <DEV>

Testing

Approved

Merge with Staging

Push <origin> <STA>

Pull <origin> <STA>

Testing

Approved

Merge with Production

Push <origin> <PRO>

Pull <origin> <DEV>

Testing

Approved

Merge with Staging

Push <origin> <STA>

Pull <origin> <STA>

Testing

Approved

Merge with Production

Push <origin> <PRO>

Pull <origin> <PRO>

Tag <TAG>

Live

LOCAL

DEVELOPMENT

STAGING

PRODUCTION

FUCK YOU GIT!

```
# Eliminar archivo/directorio
$ git rm <file_or_dir>

# Renombrar/Mover archivo/directorio
$ git mv <source> <destination>

# Deshacer `commit`
$ git commit -amend

# Deshacer `add`
$ git reset HEAD <file_or_dir>

# Recuperar la última versión
$ git checkout -- <file>

# Recuperar la última versión
$ git rebase <branch>

# Revertir `commit`
$ git revert <commit>

# Limpia el repositorio
$ git clean -df <file_or_dir>
```

LINKS

DOCUMENTACIÓN

http://git-scm.com/documentation

http://www.atlassian.com/git/tutorial/git-basics

IDE

http://www.sourcetreeapp.com/

SOCIAL CODING

https://github.com/

https://bitbucket.org/

VIDEOS

https://www.youtube.com/watch?v=ZDR433b0HJY

https://www.youtube.com/watch?v=4XpnKHJAok8

figment

grazie amici miei... dubbi?