# Git y Github

Roberto Amador

@RobertoAmador

# ¿Qué es git?

Git es un sistema de control de versiones (Version Control System [VCS]) diseñado por Linus Torvalds.

Otros sistemas parecidos son Mercurial y Subversion (SVN).

Git es el sistema de versiones más usado y popular del mundo.

Git te permite trabajar en equipo de una manera organizada y mantiene tu código protegido de casi cualquier peligro.

# Github

Github es un servicio web para almacenar repositorios de git.

Github a diferencia de git cuenta con una interfaz gráfica (Graphical User Interface [GUI]) que “simplifica” el uso de git.

Github tiene 9 millones de usuarios y 21.1 millones de repositorios, lo que lo convierte en el mayor repositorio de codigo fuente en el mundo.

# Crear un nuevo repositorio

Para crear un nuevo repositorio de git podemos utilizar

```
git init
```

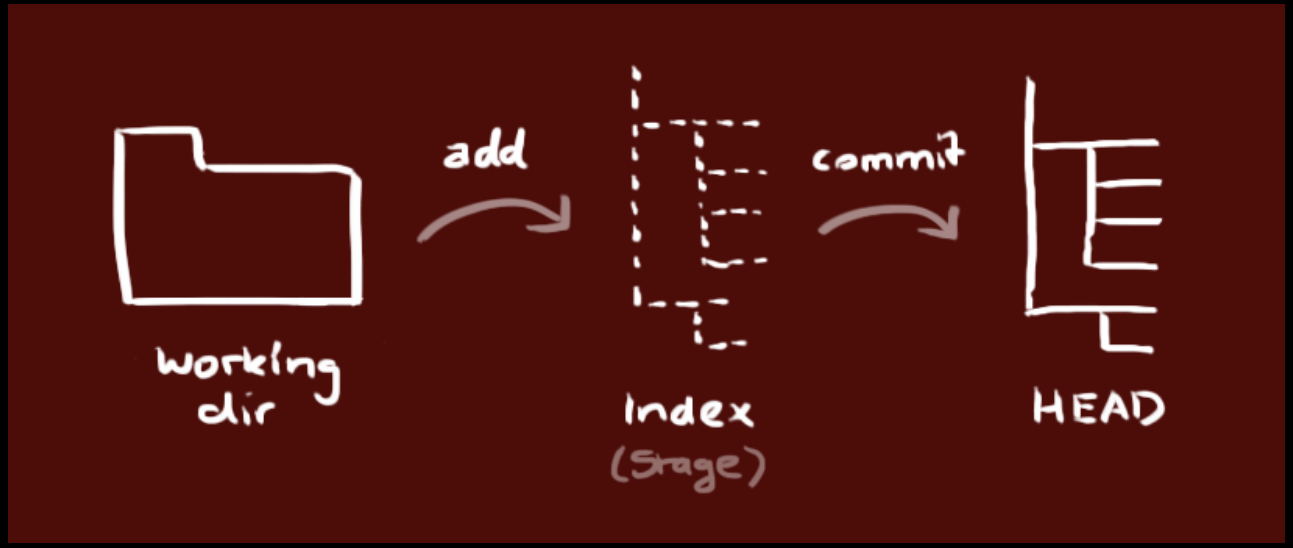
O podemos descargar un repositorio ya creado en github usando

```
git clone /path/to/repository
```

# Workflow

Un repositorio local cuenta con 3 “árboles” .

- El primero es el Working Directory y guarda tus archivos
- El segundo es Index que es el staging area
- El tercero es HEAD que apunta a el último commit hecho



# add & commit

Podemos proponer cambios (agregarlos al staging area) usando el comando

git add <filename>

git add *

Y para guardar los cambios tenemos que hacer un commit

git commit -m "Commit message“

Con esto tus cambios estan guardados en el HEAD, pero no en tu repositorio remoto

# pushing changes

Ahora que tus cambios están en el HEAD debemos pasar los cambios a tu repositorio remoto

```
git push origin master
```

Si no usaste clone para iniciar tu repositorio puedes agregar un repositorio remoto

```
git remote add origin <server>
```

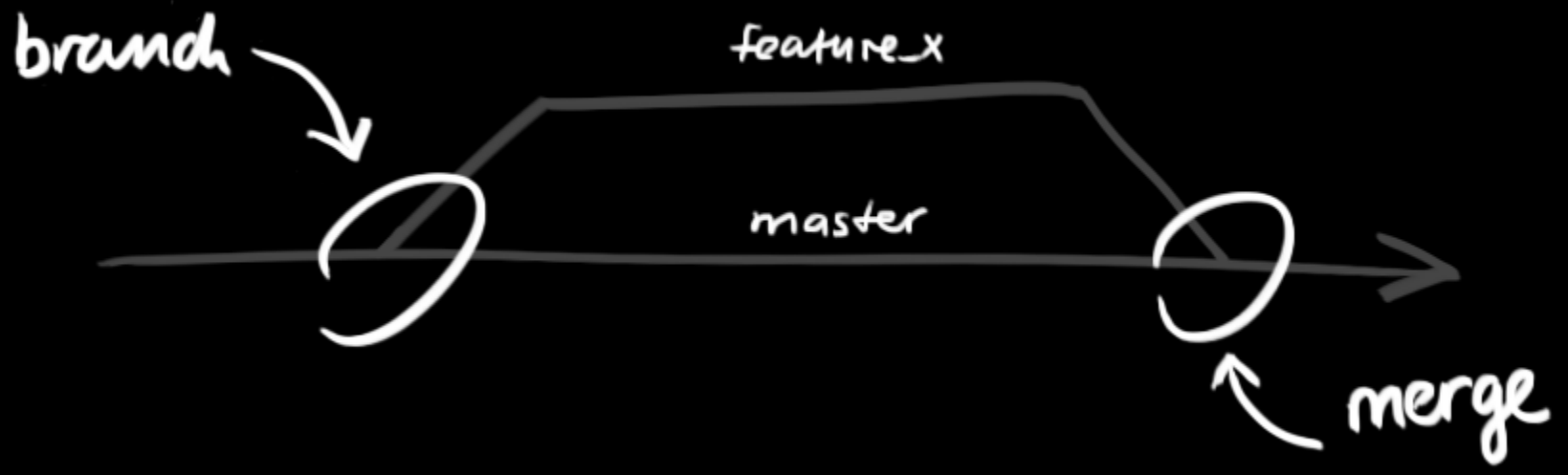
# branching

Los branches es una de las mejores funcionalidades de git. Con ella podemos crear copias de nuestro código para hacer cambios sin modificar el código base.

La branch principal se llama *master* y es la branch principal.

Al terminar de trabajar sobre un branch podemos unirla con la master

# branching



# branching

Para crear una branch llamada “feature_x” y cambiar a ella usamos

```
git checkout -b feature_x
```

Para regresar a master

```
git checkout master
```

Para borrar una branch

```
git branch -d feature_x
```

Una branch no esta disponible para los demás a menos que le hagas un push a repositorio remoto

```
git push origin <branch>
```

# pull & merge

Para actualizar un repositorio remoto

```
git pull
```

Para unir una branch con otra. Ejemplo: master con otra branch

```
git merge <branch>
```

# pull & merge

Git va a tratar de unir los cambios de manera automática pero no siempre sucede sin encontrar problemas.

Por ejemplo: Cuando dos personas modifican el mismo archivo en la misma linea git puede tener problemas para saber que cambios son los que debe de mantener.

En estos casos se debe de hacer los cambios de manera manual y agregarse con

```
git add <filename>
```

# tagging

Es recomendado agregar tags cuando se trata de un release para tu software.

Por ejemplo podemos agregar el tag 1.0.0 usando

```
git tag 1.0.0 1b2e1d63ff
```

*1b2e1d63ff* es el commit id del commit que queremos tagear.

# log

Para ver los logs con los ids de los commits

```
git log
```

Para tener un log de una manera más visual

```
git log --graph --oneline --decorate --all
```

# Ejercicio: Git basics

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

# Pro tips

Haz un commit para cosas relacionadas. Si arreglas dos bugs entonces deberían de haber dos commits.

Haz un commit seguido.

Nunca hagas un commit con código a medio acabar o sin probarlo.

Escribe buenos mensajes para tus commits. Empieza con 50 caracteres como resumen minimo.

Usa los branches!!

# Ejercicio: Git branching

http://pcottle.github.io/learnGitBranching/

# Pro tips

Para usar el GUI default

```
gitk
```

Mostrar los outputs con colores

```
git config color.ui true
```

Mostrar los logs en una linea por default

```
git config format.pretty oneline
```

Para agregar de manera interactiva

```
git add -i
```

# Recursos

http://rogerdudler.github.io/git-guide/

http://www.git-tower.com/blog/git-cheat-sheet/

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

https://help.github.com/articles/good-resources-for-learning-git-and-github/

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