

---

# Git

*Una de las claves de trabajar como un profesional es tener un entorno profesional.*

José A. Reyes H.  
@josereyes\_ah





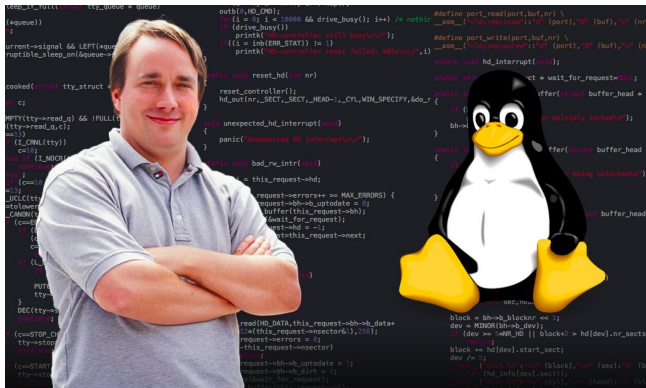
# ¿Qué es Git?



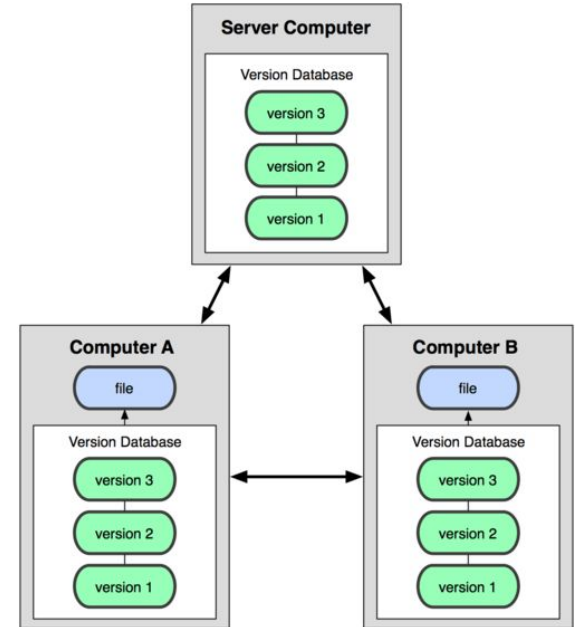
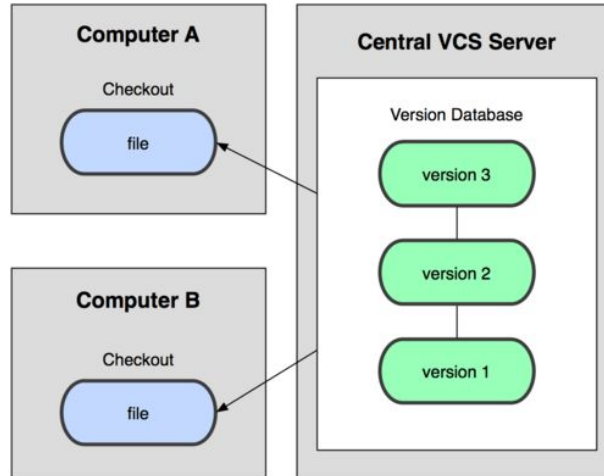
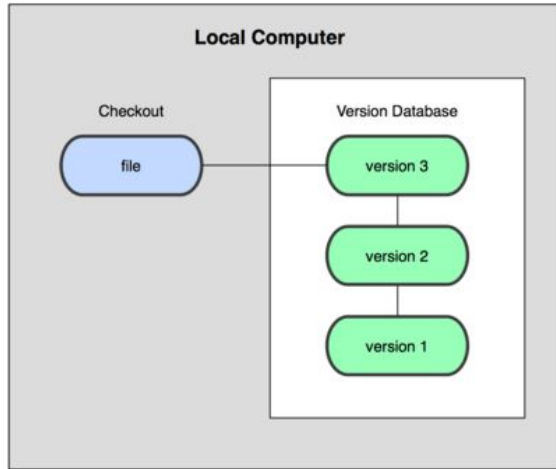
Es un software de control de versiones (VCS).  
Nace en 2005.

VCS es la administración de los diversos cambios que se realizan sobre los elementos de algún producto. Esto facilita la gestión de las distintas versiones de cada producto desarrollado.

Se utiliza principalmente en la industria informática para controlar las distintas versiones del código fuente.



# Control de versiones



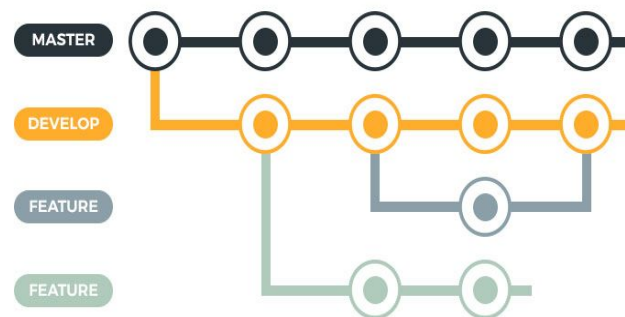
Sistemas de control de versiones locales.

Sistemas de control de versiones centralizados.

Sistemas de control de versiones distribuidos.

# Características de Git.

1. Velocidad: Tremendamente rápido.
2. Diseño: sencillo y fácil de usar.
3. Fuerte apoyo al desarrollo no lineal (tiene un increíble sistema de ramificación (**branching**)).
4. Capaz de manejar grandes proyectos (como el núcleo de **Linux**) de manera eficiente (velocidad y tamaño de los datos)



# ¿Qué nos aporta git?

- **Auditoría del código:** saber quién ha modificado los archivos.
- Control sobre cómo ha cambiado nuestro proyecto con el paso del tiempo.
- Control de versiones a través de etiquetas: versión 1.0, versión 1.0.1, etc.
- **Seguridad:** todas las estructuras internas de datos están firmadas con **SHA-1**.
- Mejora nuestra capacidad de trabajar en equipo.
- Es útil para cualquier profesión.



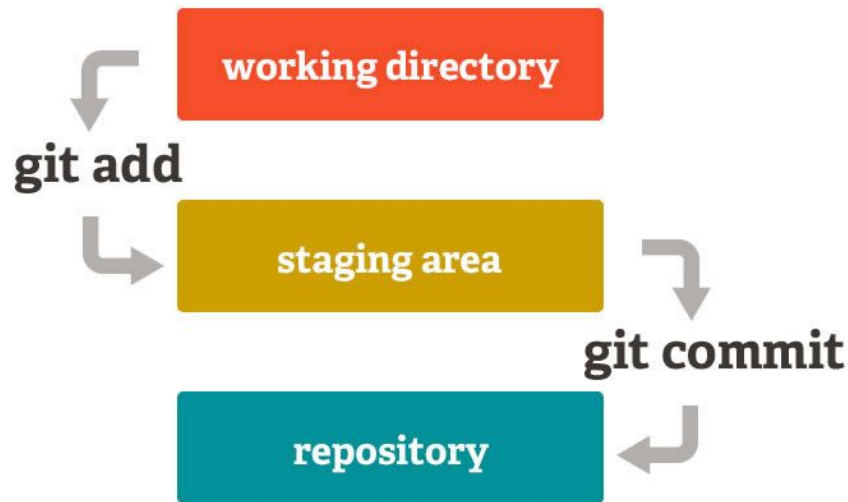
# Flujo de trabajo

Git tiene 3 estados principales:

1. Confirmado (committed),
2. Modificado (modified), y
3. Preparado (staged).

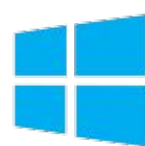
El flujo de trabajo básico en Git es algo así:

1. **Modificas** una serie de archivos en tu directorio de trabajo.
2. **Preparas** los archivos, añadiéndolos a tu área de preparación.
3. **Confirmas** los cambios, lo que toma los archivos tal y como están en el área de preparación, y almacena en tu directorio de Git.



# Versión, instalación y código fuente.

- Version 2.9.3 - 2.13
- Descargar:
  - **MS**: <https://git-for-windows.github.io/>
  - **OSX**: <http://git-scm.com/download/mac>
  - **Linux**. # dnf install git-core



Windows



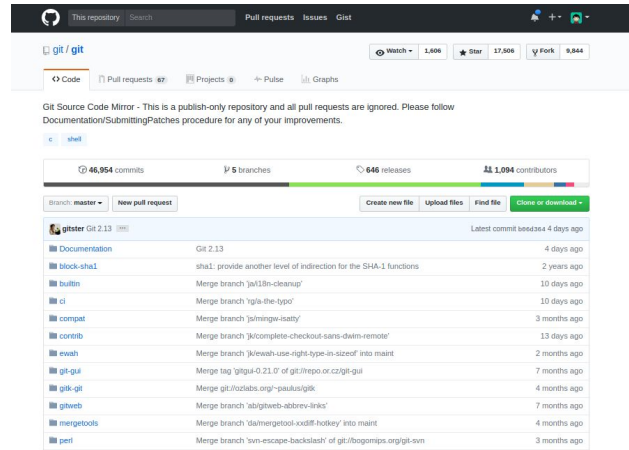
Mac



Linux

## Git Source Code Mirror:

- <https://github.com/git/git/>



# Configuración de Git.

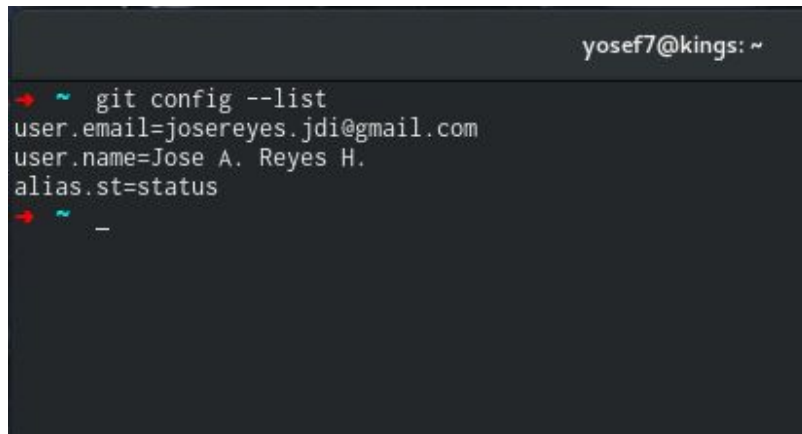
- Configuramos nuestro usuario y correo electrónico.

```
$ git config --global user.name "firstname lastname"
```

```
$ git config --global user.email "valid-email"
```

- Comprobamos la configuración actual

```
$ git config --list
```

A terminal window with a dark background. The prompt is 'yosef7@kings: ~'. The command 'git config --list' has been executed, and the output is displayed in a monospaced font. The output shows three lines: 'user.email=josereyes.jdi@gmail.com', 'user.name=Jose A. Reyes H.', and 'alias.st=status'. There is a red arrow cursor on the line following the output.

```
yosef7@kings: ~  
➔ ~ git config --list  
user.email=josereyes.jdi@gmail.com  
user.name=Jose A. Reyes H.  
alias.st=status  
➔ ~ -
```



# Comandos básicos.

- Crear un nuevo repositorio local  
\$ git **init** nombred carpeta [pract-git]
- Verificar el estado de los archivos  
\$ git **status**
- Crear un archivo .txt  
\$ vim prueba00.txt
- Registrar los cambios usando el comando.  
\$ git **add** prueba.txt (git add .)
- Se entregan todos los cambios.  
\$ git **commit** -m "Se añade archivo txt"

# Comandos básicos.

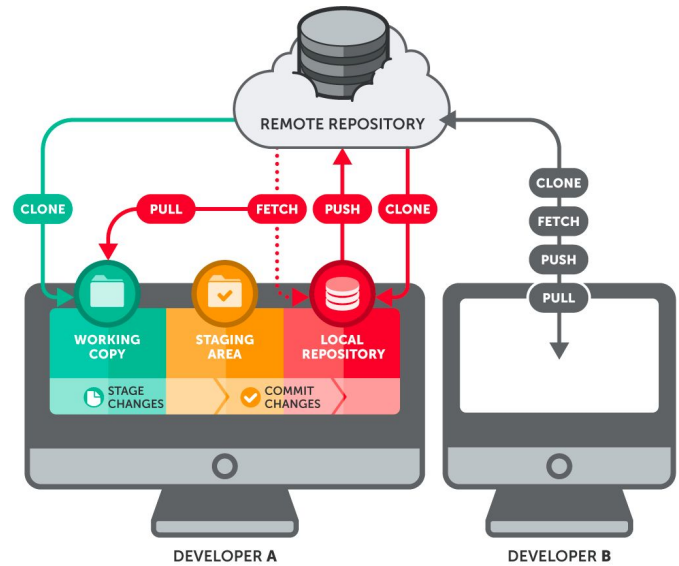
- Crear varios archivos txt (01-02)  
\$ git *add* '\*.txt'
- \$ git *commit* -m 'Añaden archivos txt'

# GitHub

Crear una cuenta en GitHub.

GitHub es un servicio que te permite guardar repositorios de GIT en la nube.

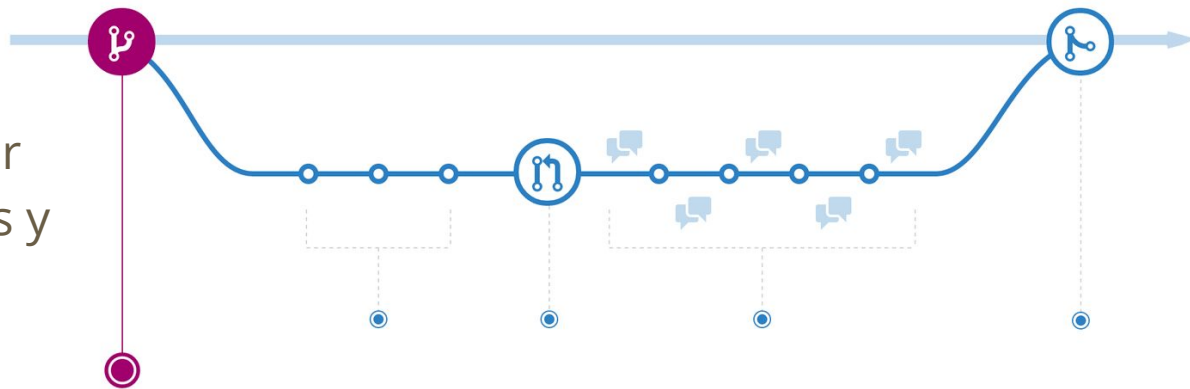
- `git@github.com:yosef7/prueba.git`
- `git remote add origin`  
`git@github.com:yosef7/prueba.git`



# Comandos básicos.

- \$ git **push** -u origin master (subir a la nube)
  - El argumento **-u** sirve para que GIT recuerde los parámetros (origin master) y las siguientes veces puedas utilizar simplemente el comando 'git push'.

- \$ git **pull** origin master (Descarga los cambios y los combina).



# Temas Avanzados



**git**

# Otros comandos

- Muestra la bitácora de commits  
\$ git log  
\$ git log -n3
- Ver quien cambio el archivo y cuando.  
\$ git blame [nombrearchivo]
- Elimina el archivo  
\$ git rm archivo.txt
- Etiquetas  
\$ git tag 1.0.0 rb2e1d63ff

# Buenas prácticas.

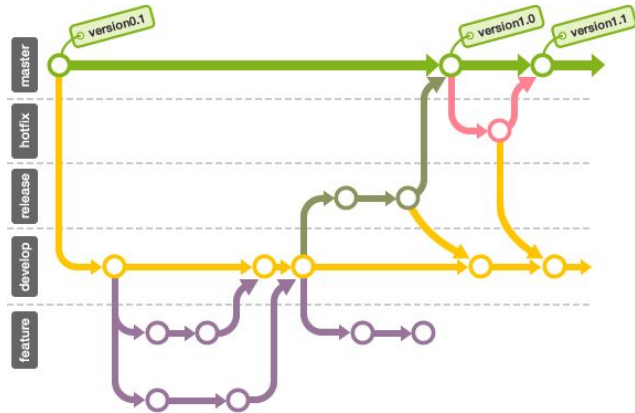
Una manera de trabajar con GIT, es establecer un patrón de trabajo a nivel de empresa.

Se utilizan 4 tipos de ramas:

- a. Master.
- b. Development
- c. Features
- d. Hotfix

# Tipos de ramas

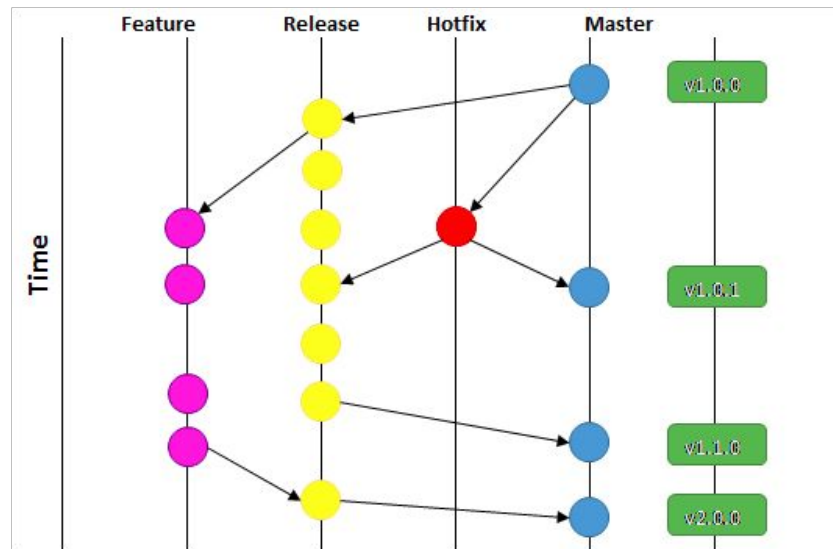
1. **Master:** rama principal. Contiene el repositorio que se encuentra publicado en producción, por lo que debe estar siempre estable.
2. **Development:** rama copia de la master.
3. **Features:** cada nueva funcionalidad se debe realizar en una rama nueva, se debe sacar de development.
4. **Hotfix:** son bugs que surgen en producción, por lo que deben arreglar y publicar de forma urgente.





# Más Comandos

- Listar rama existentes  
`$ git branch -av`
- Cambiar ramas en el HEAD  
`$ git checkout [rama]`
- Crear rama  
`$ git branch [nuevarama]`
- Eliminar una rama  
`$ git branch -d [rama]`



# Buenas prácticas

1. Hacer ***commit*** de cambios relacionados.
2. Hacer ***commit*** con mayor frecuencia.
3. Probar el código antes de hacer ***commit***.
4. Escribir un mensaje entendible en el ***commit***.
5. Utilizar ***ramas***.
6. Seguir un flujo de trabajo.





¿Quién usa Git?

Google

facebook



Microsoft



LinkedIn

NETFLIX



GNOME



eclipse



# Git



*GIT es un Ctrl-Z con esteroides.*

# Paginas de interes.

<https://www.codeschool.com/>

<https://try.github.io/>

<http://help.github.com/>

<http://rogerdudler.github.io/git-guide/index.es.html>

<https://education.github.com/git-cheat-sheet-education.pdf>

<b>-p</b>	Muestra el parche introducido en cada confirmación.
<b>--word-diff</b>	Muestra el parche en formato de una palabra.
<b>--stat</b>	Muestra estadísticas sobre los archivos modificados en cada confirmación.
<b>--shortstat</b>	Muestra solamente la línea de resumen de la opción --stat.
<b>--name-only</b>	Muestra la lista de archivos afectados.
<b>--name-status</b>	Muestra la lista de archivos afectados, indicando además si fueron añadidos, modificados o eliminados.
<b>--abbrev-commit</b>	Muestra solamente los primeros caracteres de la suma SHA-1, en vez de los 40 caracteres de que se compone.
<b>--relative-date</b>	Muestra la fecha en formato relativo (por ejemplo, “2 weeks ago” (“hace 2 semanas”)) en lugar del formato completo.
<b>--graph</b>	Muestra un gráfico ASCII con la historia de ramificaciones y uniones.
<b>--pretty</b>	Muestra las confirmaciones usando un formato alternativo.
<b>--oneline</b>	Un cómodo acortamiento de la opción --pretty=oneline --abbrev-commit.

# Configuraciones especiales

Colores especiales para la consola

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

Mostrar sólo una línea por cada commit en la traza

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

Agregar archivos de forma interactiva

```
$ git add -i
```

# Gracias!

@josereyes\_ah