Curso

Sistema de control de versiones

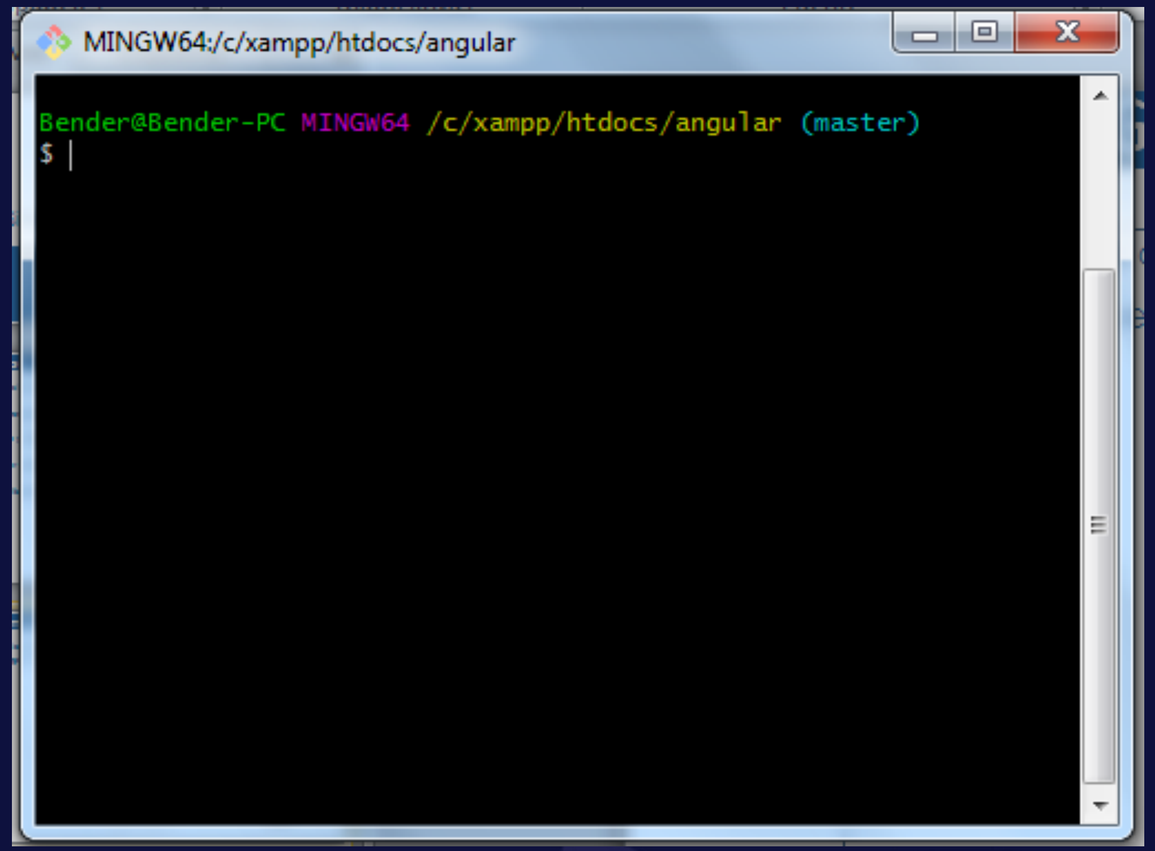
Bernabé
Gutiérrez
Rodríguez

git

Contenido

Instalación de Git

1. Ir al sitio oficial de Git: https://git-scm.com
2. Descargar la versión más reciente e instalar



Introducción a Git

Git es un sistema de control de versiones open source el cual permite crear equipos de trabajo colaborativo para organizar nuestros poryectos de software de manera sencilla.

VS

Introducción a Git

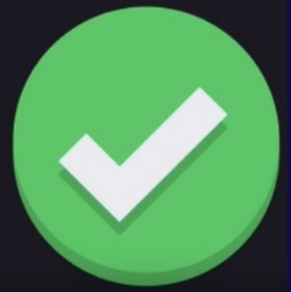
Componentes de Git

Working directory

Aquí es donde editamos y trabajamos con nuestros proyectos.

Staging area

Aquí es donde escogemos que archivos están listos para pasar al tercer estado, al igual que decidimos que archivos no están listos por el momento.



Flujo de trabajo

1 → 2 → 3

1. Modificar, crear, editar una serie de archivos.
2. Escoger los archivos que están listos.
3. Registro de nuestro proyecto.

Primeros pasos

Configurar usuario y correo electrónico

Comandos:

```
git config --globlal user.name "Bernabe Gutierrez"
git config --globlal user.mail "bernita.gutierrez@gmail.com"
git config --globlal color.ui true
```

git help :Ayuda de git

q : exit

Primeros pasos

Iniciando con un proyecto Git

Comandos:

`git init` : marca el inicio de nuestro proyecto Git

`git status` : estado de nuestro proyecto

`git add -A` : Marcamos los archivos que están listos para el siguiente paso

`git commit -m "mensaje"` : guarda los cambios con un mensaje para identificarlos

`git log` :lista de todos nuestros commist con su info

`git log > commits.txt :`descarga la lista de todos nuestros commist con su info en un archivo txt

Primeros pasos

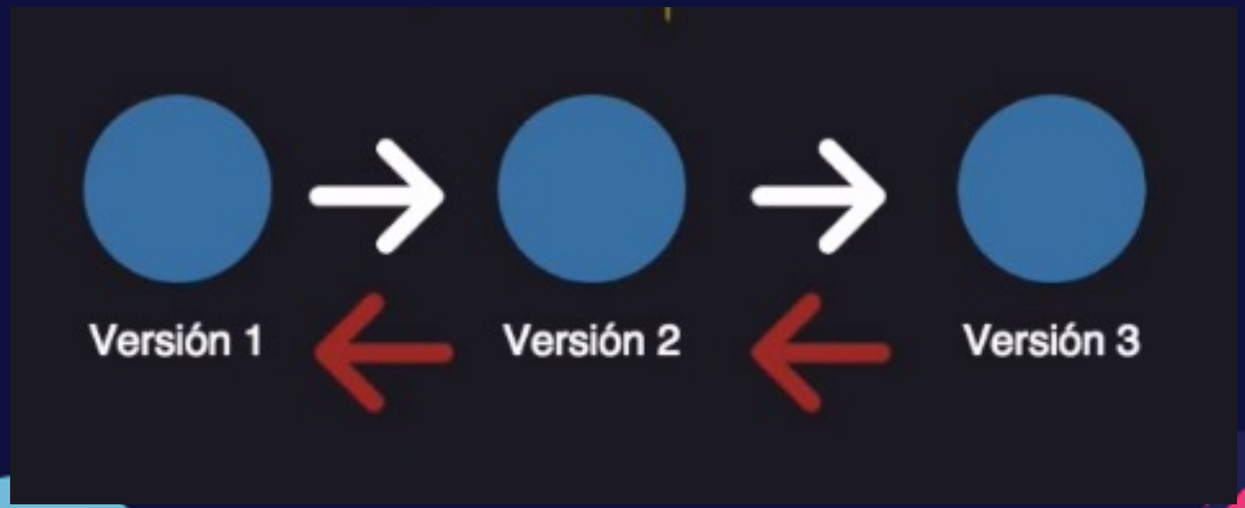
Cambiar (Viajar) entre commits

Comandos:

`git checkout name_branch :` viajamos a través de los commist o ramas

`git checkout master :` regresamos a la rama principal

`git reset key_commit :` Elimina los commits



Primeros pasos

Eliminar commit

Comandos:

`git reset --soft codigo_commit` : elimina el commit pero no elimina el codigo

`git reset --hard codigo_commit` : elimina el commit juntamente con el codigo :(

Ramas

Conceptos básicos

Head es en el commit donde nos encontramos

Rama Master

La rama master es en donde comenzamos a trabajar, es la rama principal y estable de nuestro proyecto.

Ramas

Es una linea de tiempo en nuestro proyecto, que nos sirven para arreglar errores, experimentar, hacer grandes cambios, etc.

Fusiones

Es la creación de un nuevo **commit** juntando una rama con otra.

Ramas

Creación de ramas

`git branch` : muestra las ramas existentes

`git branch name_branch` : crea una rama

`git checkout name_branch` : cambiar de rama

`git checkout -b name_branch` : crea una rama y te posiciona en ella

`git branch -D name_branch` : elimina una rama

`git branch -a` : muestra las ramas ocultas

Merge

Merge

para hacer merge, debe posicionarse en la rama master

`git checkout master`

luego hacer merge a la rama que se va a unir a master

`git merge name_branch_sec`

Ramas y Merge

Tipos de Merge

Fast-Forward

Solo va a hacer la fusión, esto pasa normalmente cuando se trabaja con archivos diferentes o líneas de código distintas.

Manual Merge

Antes de hacer la fusión tiene que pasar por nosotros, normalmente ocurre cuando se trabaja en los mismos archivos o líneas de códigos.

Consideraciones iniciales

GitLab

GitHub

≠

git

GitLab

Primeros pasos

git remote add origin GitHub_key : inicia conexión entre repo local y GitHub

git remote -v :Verificar si existe una conexión entre el repo local y GitHub

git remote remove :Elimina la conexión entre el repo local y GitHub

git push origin master :(o rama) carga el proyecto a GitHub

git push origin master -f :(o rama) carga el proyecto a GitHub

git push origin v1.0 : sube a GitHub un tag

$ git push origin --tags : sube a GitHub todos los tags

Git Clone & Git Push

Otras características importantes

```
$ git clone
```

Git clone no sirve para clonar un proyecto, normalmente, se usa cuando no nos interesa colaborar en el proyecto.

```
$ git push
```

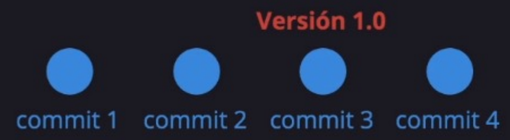
Git push manda nuestros cambios (commits) a Github.

```
$ git push origin master
```

Tags

Tipos de tag

Tags (Etiquetas)



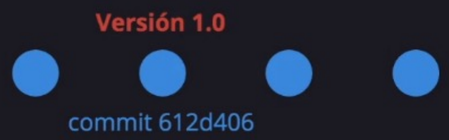
Los **tags** son simples puntos específicos en la historia de nuestro proyecto y se usan para marcar alguna versión del mismo.

Tags anotadas

```
$ git tag -a v1.0 -m "Mensaje"
```

Las **tags anotadas** son almacenadas como objetos completos dentro de la base de Git y contienen más información.

```
$ git tag -a v1.0 -m 'Mensaje' 612d406
```



Al agregar el código **SHA** podemos especificar donde se va a aplicar una etiqueta.

Workflows

Primeros pasos

- crear una nueva organización desde GitHub
- invitar a colaboradores
- darles permisos
- desde el repositorio ir a setings y mover el repositorio a la organización creada

Workflows

Comandos

`git clone` : colaborador copia repositorio la su pc

`git remote remove origin` :el admin debe eliminar la conexión del repo remoto y hacer referencia al repo de la organización

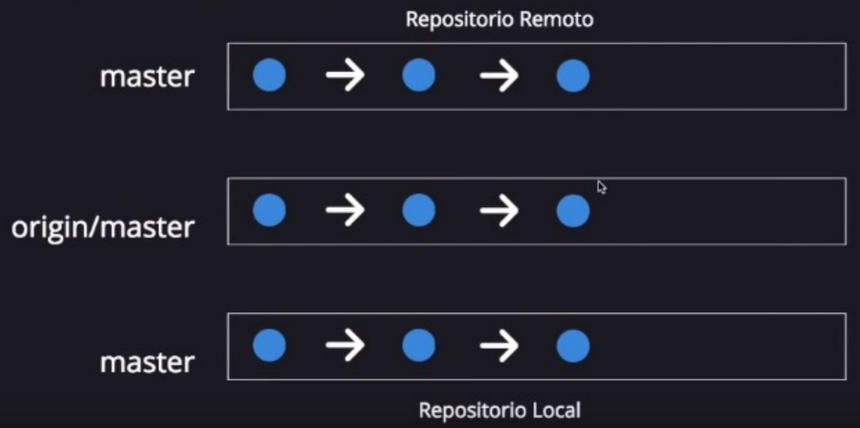
`git remote add origin lik_github` : crear nuevamente la conexión al repo remoto pero de la organización

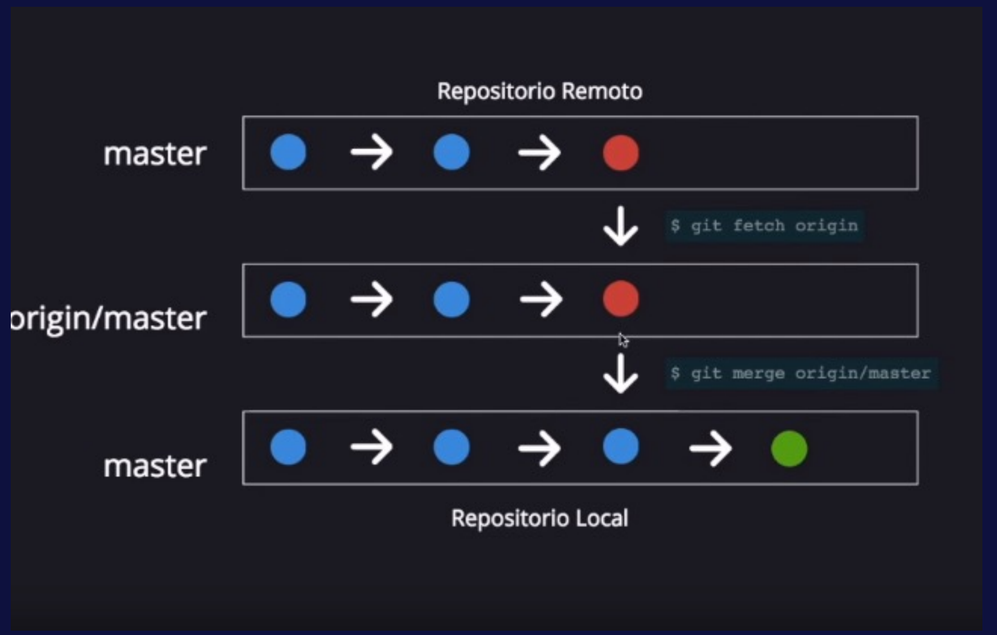
Repetir por cada usuario cada vez que haya cambios en el repo remoto
`git fetch origin` : Actualizar el repo local con el repo remoto

`$ git merge origin/master` unir los commits locales con los remotos en la rama master de cada usuario

Resumen

Cuando nosotros conectamos el **repositorio local** con el **repositorio remoto**, hay una rama oculta y prácticamente es un espejo entre estos dos (Idéntico a master).





Git Ignore

- Un **`gitignore`** es archivo que especifica archivos sin seguimiento intencional que Git debe ignorar. Los archivos ya rastreados por Git no se ven afectados.
- Ejemplo

```
Helo.* (Ignora a hello.txt, hello.c pero no a d/hello.jav)
/nbproject/ (Ignora todos los elementos del directorio nbproject)
/docs/database.sql (ignora el archivo database.sql)
/dompdf/
*.com
*.zip
*.sql
.DS_Store
.Thumbs.db
```

Referencia: https://git-scm.com/docs/gitignore

Gracias

Fuente https://git-scm.com/