



UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

ESCUELA DE ESTUDIOS DE POSTGRADO

FACULTAD DE INGENIERÍA

SEMINARIO I

José Pablo Recinos Guzmán	3395402302101	999010792
---------------------------	---------------	-----------

¿Qué es GIT?

GIT es un sistema de control de versiones distribuido que permite registrar los cambios que se realizan en los archivos y volver a versiones anteriores si algo sale mal. Fue diseñado por Linus Torvalds para garantizar la eficiencia y confiabilidad del mantenimiento de versiones. GIT también proporcionar un listado de los cambios(commits) y se puede volver atrás en el tiempo a cualquiera de esos cambios o commits. Además nos ofrece la posibilidad de trabajar con ramas de desarrollo, que nos van a permitir desarrollar cosas que divergen mucho del programa principal.

GIT nos ofrece:

- Rendimiento
- Seguridad
- Flexibilidad



Control de versiones con GIT

Un Sistema de Control de Versiones se refiere al método utilizado para guardar las versiones de un archivo para referencia futura.

De manera intuitiva muchas personas ya utilizan control de versiones en sus proyectos al renombrar las distintas versiones de un mismo archivo de varias formas como blogScript.js, blogScript_v2.js, blogScript_v3.js, blogScript_final.js, blogScript_definite_final.js, etcétera. Pero esta forma de abordarlo es propenso a errores e inefectivo para proyectos grupales.

Además, con esta forma de abordarlo, rastrear qué cambió, quién lo cambió y por qué se cambió, es un esfuerzo tedioso. Esto resalta la importancia de un sistema de control de versiones confiable y colaborativo como Git.



Estados de un archivo en GIT

- Estado Modificado

Un archivo en el estado modificado es un archivo revisado – pero no acometido (sin registrar). En otras palabras, archivos en el estado modificado son archivos que se han modificado, pero no se ha instruido explícitamente a Git que controle.

- Estado Preparado

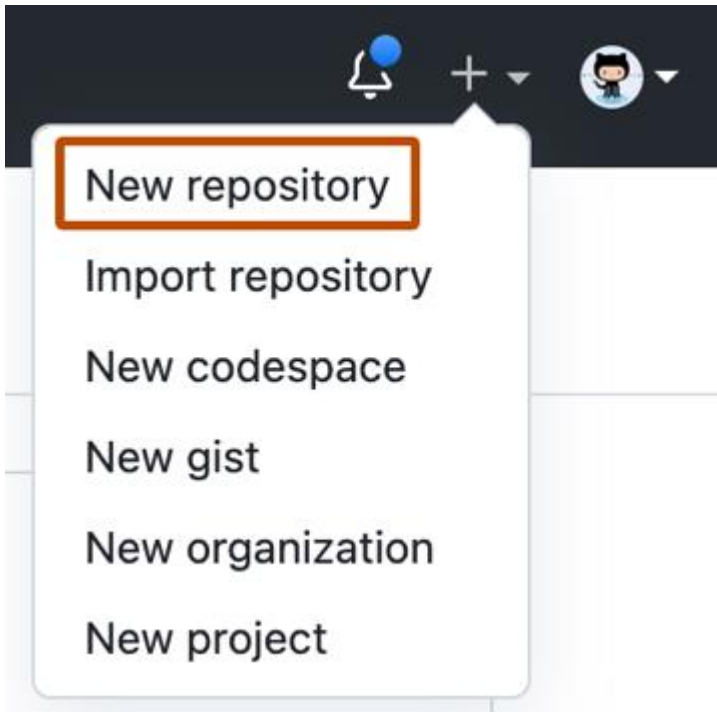
Archivos en la etapa preparado son archivos modificados que han sido seleccionados – en su estado (versión) actual – y están siendo preparados para ser guardados (acometidos) al repositorio git durante la próxima instantánea de confirmación. Una vez que el archivo está preparado implica que has explícitamente autorizado a Git que controle la versión de ese archivo.

- Estado Confirmado

Archivos en el estado confirmado son archivos que se guardaron en el repositorio .git exitosamente. Por lo tanto, un archivo confirmado es un archivo en el cual se ha registrado su versión preparada en el directorio (carpeta) Git.


Configuración de un repositorio en GIT

En la esquina superior derecha de cualquier página, se utiliza el menú desplegable y se selecciona **New repository** (Nuevo repositorio).



Se escribe el nombre del repositorio

Owner * Repository name *

 octocat ▾ / ✓

Great repository names are short and memorable. Need inspiration? How about [vigilant-meme?](#)

Description (optional)

Opcionalmente, se puede agregar una descripción del repositorio. Por ejemplo, "Mi primer repositorio en GitHub".

Se elige la visibilidad del repositorio. Se selecciona **Initialize this repository with a README** (Inicializar este repositorio con un archivo Léame). Y se hace clic en **Create repository** (Crear repositorio).

Comandos en GIT

Configuración Básica

Configurar Nombre que salen en los commits

```
git config --global user.name "dasdo"
```

Configurar Email

```
git config --global user.email dasdo1@gmail.com
```

Marco de colores para los comando

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

Iniciando repositorio

Iniciamos GIT en la carpeta donde esta el proyecto

```
git init
```

Clonamos el repositorio de github o bitbucket

```
git clone <url>
```

Añadimos todos los archivos para el commit

```
git add .
```

Hacemos el primer commit

```
git commit -m "Texto que identifique por que se hizo el commit"
```

subimos al repositorio

```
git push origin master
```

GIT CLONE

Clonamos el repositorio de github o bitbucket

```
git clone <url>
```

Clonamos el repositorio de github o bitbucket ?????

```
git clone <url> git-demo
```

GIT ADD

Añadimos todos los archivos para el commit

```
git add .
```

Añadimos el archivo para el commit

```
git add <archivo>
```

Añadimos todos los archivos para el commit omitiendo los nuevos

```
git add --all
```

Añadimos todos los archivos con la extensión especificada

```
git add *.txt
```

Añadimos todos los archivos dentro de un directorio y de una extensión específica

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

Añadimos todos los archivos dentro de un directorios

```
git add docs/
```

GIT COMMIT

Cargar en el HEAD los cambios realizados

```
git commit -m "Texto que identifique por que se hizo el commit"
```

Agregar y Cargar en el HEAD los cambios realizados

```
git commit -a -m "Texto que identifique por que se hizo el commit"
```

De haber conflictos los muestra

```
git commit -a
```

Agregar al ultimo commit, este no se muestra como un nuevo commit en los logs.
Se puede especificar un nuevo mensaje

```
git commit --amend -m "Texto que identifique por que se hizo el commit"
```

GIT PUSH

Subimos al repositorio

```
git push <origien> <branch>
```

Subimos un tag

```
git push --tags
```

GIT LOG

Muestra los logs de los commits

```
git log
```

Muestras los cambios en los commits

```
git log --oneline -stat
```

Muestra graficos de los commits

```
git log --oneline --graph
```

GIT DIFF

Muestra los cambios realizados a un archivo

```
git diff  
git diff --staged
```

GIT HEAD

Saca un archivo del commit

```
git reset HEAD <archivo>
```

Devuelve el ultimo commit que se hizo y pone los cambios en staging

```
git reset --soft HEAD^
```

Devuelve el ultimo commit y todos los cambios

```
git reset --hard HEAD^
```

Devuelve los 2 ultimo commit y todos los cambios

```
git reset --hard HEAD^^
```

Rollback merge/commit

```
git log  
git reset --hard <commit_sha>
```

GIT REMOTE

Agregar repositorio remoto

```
git remote add origin <url>
```

Cambiar de remote

```
git remote set-url origin <url>
```

Remover repositorio

```
git remote rm <name/origin>
```

Muestra lista repositorios

```
git remote -v
```

Muestra los branches remotos

```
git remote show origin
```

Limpiar todos los branches eliminados

```
git remote prune origin
```

GIT BRANCH

Crea un branch

```
git branch <nameBranch>
```

Lista los branches

```
git branch
```

Comando -d elimina el branch y lo une al master

```
git branch -d <nameBranch>
```

Elimina sin preguntar

```
git branch -D <nameBranch>
```

GIT TAG

Muestra una lista de todos los tags

```
git tag
```

Crea un nuevo tags

```
git tag -a <verison> - m "esta es la versión x"
```

GIT REBASE

Los rebase se usan cuando trabajamos con branches esto hace que los branches se pongan al día con el master sin afectar al mismo

Une el branch actual con el mastar, esto no se puede ver como un merge

```
git rebase
```

Cuando se produce un conflicto no das las siguientes opciones:

cuando resolvemos los conflictos --continue continua la secuencia del rebase donde se pauso

```
git rebase --continue
```

Omite el conflicto y sigue su camino

```
git rebase --skip
```

Devuelve todo al principio del rebase

```
git reabse --abort
```

Para hacer un rebase a un branch en especifico

```
git rebase <nameBranch>
```


OTROS COMANDOS

Lista un estado actual del repositorio con lista de archivos modificados o agregados

```
git status
```

Quita del HEAD un archivo y le pone el estado de no trabajado

```
git checkout -- <file>
```

Crea un branch en base a uno online

```
git checkout -b newlocalbranchname origin/branch-name
```

Busca los cambios nuevos y actualiza el repositorio

```
git pull origin <nameBranch>
```

Cambiar de branch

```
git checkout <nameBranch/tagname>
```

Une el branch actual con el especificado

```
git merge <nameBranch>
```

Verifica cambios en el repositorio online con el local

```
git fetch
```

Borrar un archivo del repositorio

```
git rm <archivo>
```

Fork

Descargar remote de un fork

```
git remote add upstream <url>
```

Merge con master de un fork

```
git fetch upstream  
git merge upstream/master
```