Entornos de Desarrollo

UD4 – Optimización y Documentación

Actividad 0

# Actividad 1. Uso básico de Git

Requisitos:

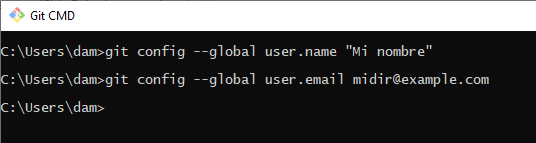
* Git.

## Configuración de usuario de Git

1. Lo primero que debemos hacer para utilizar Git es establecer nuestro nombre de usuario y dirección de correo electrónico. Esta información se asociará a los cambios que confirmemos en los repositorios. Debemos ejecutar los comandos en la consola de Git:

git config --global user.name "John Doe"

git config --global user.email johndoe@example.com



## Actividad con un repositorio local.

Preparación:

* Creamos un directorio donde guardaremos todos los repositorios, lo llamamos repos

mkdir repos

* Dentro de repos creamos el directorio de trabajo (working dir) para nuestro repositorio
* y añadimos algunos ficheros con texto (no necesariamente código).

mkdir repos/<working dir>

cd repos/<working dir>

Creación de repositorio:

git init

*Alternativamente podríamos clonar un repositorio existente con*

*git clone <URL>*

Añadimos los ficheros a stage para hacerles seguimiento y comprobamos el estado

git add .

git status

Hacemos commit de los ficheros a la rama master

git commit -m “Commit inicial”

Ahora modificamos alguno de los ficheros y lo añadimos a stage.

git add <fichero>

Actualizamos master

git commit “Segundo commit de prueba”

Creamos una nueva rama, comprobamos que ramas tenemos y cambiamos a la nueva

git branch <rama>

git branch -l

## git checkout <rama>

## Actividad: sincronización con un repositorio remoto.

git clone https://github.com/githubtraining/hellogitworld.git

*Alternativamente si ya tenemos un repositorio podemos añadir un repositorio remoto y obtener el código*

*git remote add origin https://github.com/githubtraining/hellogitworld.git*

git pull

Si estamos autorizados para actualizar el repositorio remoto

git push <origien> <branch>

Para esta práctica vamos a crear un repositorio en local para usarlo como remoto.

mkdir repos/remoto.git

cd repos/remoto.git

git init --bare

Asociamos el repositorio remoto y subimos nuestro código

cd repos/<working dir>

git remote add origin <usuario>@localhost:repos/remoto.git

git push origin master

## [Comandos git](https://gist.github.com/dasdo/9ff71c5c0efa037441b6)

## **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 está el proyecto, clonamos repositorio, añadimos archivos (todos), hacemos primer commit y lo subimos al repositorio remoto

git init

git clone <url>

git add .

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

git push origin master

## **GIT CLONE**

Clonamos el repositorio

git clone <url>

Clonamos el repositorio

git clone <url> git-demo

## **GIT ADD**

Añadimos archivos a stage para el commit

git add . *todos los archivos*

git add <archivo> *un archivo*

git add --all *todos los archivos para el commit omitiendo los nuevos*

git add \*.txt *todos los archivos con la extensión especificada*

git add docs/\*.txt *archivos dentro de un directorio con una extensión*

git add docs/ *todos los archivos dentro de un directorios*

Reiniciamos stage, eliminamos de stage los archivos

git reset

## **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 último 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 gráficos de los commits

git log --oneline --graph

## **GIT DIFF**

Muestra los cambios realizados a un archivo

git diff

git diff --staged

## **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"

## ***Otros comandos avanzados.***

## **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 master, 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>

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