GIT

Comandos básicos

Lun, 04/29/2013 - estebanvalerioh

Lista de Comandos en GIT (github)

Contenido

git help

Muestra una lista con los comandos más utilizados en GIT.

git init

Podemos ejecutar ese comando para crear localmente un repositorio con GIT y así utilizar todo el funcionamiento que GIT ofrece. Basta con estar ubicados dentro de la carpeta donde tenemos nuestro proyecto y ejecutar el comando. Cuando agreguemos archivos y un commit, se va a crear el branch master por defecto.

git add + path

Agrega al repositorio los archivos que indiquemos.

git add -A

Agrega al repositorio TODOS los archivos y carpetas que estén en nuestro proyecto, los cuales GIT no está siguiendo.

git commit -m "mensaje" + archivos

Hace commit a los archivos que indiquemos, de esta manera quedan guardados nuestras modificaciones.

git commit -am "mensaje"

Hace commit de los archivos que han sido modificados y GIT los está siguiendo.

git checkout -b NombreDeBranch

Crea un nuevo branch y automaticamente GIT se cambia al branch creado, clonando el branch desde donde ejecutamos el comando.

git branch

Nos muestra una lista de los branches que existen en nuestro repositorio.

git checkout NombreDeBranch

Sirve para moverse entre branches, en este caso vamos al branch que indicamos en el comando.

git merge NombreDeBranch

Hace un merge entre dos branches, en este caso la dirección del merge sería entre el branch que indiquemos en el comando, y el branch donde estémos ubicados.

git status

Nos indica el estado del repositorio, por ejemplo cuales están modificados, cuales no están siendo seguidos por GIT, entre otras características.

git clone URL/name.git NombreProyecto

Clona un proyecto de git en la carpeta NombreProyecto.

git push origin NombreDeBranch

Luego de que hicimos un git commit, si estamos trabajando remotamente, este comando va a subir los archivos al repositorio remoto, específicamente al branch que indiquemos.

git pull origin NombreDeBranch

Hace una actualización en nuestro branch local, desde un branch remoto que indicamos en el comando.

Lista de Comandos en 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 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 especifica

```
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
```