GIT

Instalacion en Linux (Ubuntu) y breve historia

Realizado con un curso de la plataforma de Platzi.
@autor Do it Yourself

Asumiremos que usamos un entorno Linux y tenemos conocimientos medio alto sobre Bash Shell.

¿Qúe es Git?: de forma breve es un software que nos ayuda a tener el control de nuestro código.
Desarrolado: Linus Torlvalds
Año: 2007
Lenguaje: C, Bournes Shell, Perl.
Licencia: GNU GPL v2

Más información: https://es.wikipedia.org/wiki/Git

Editor que usaremos es VS code.
Desarrollado: Microsoft
Año: 2015
Lenguaje: Javascript, Css
Licencia: Licencia MIT, binarios: Freeware

Más información: https://es.wikipedia.org/wiki/Visual_Studio_Code

---

Instalacion en linux:

1. Comprobar si ya tenemos git con el comando $git -version, sino seguimos el paso 2
2. $sudo apt-get update && sudo apt-get upgrade && sudo apt-get install git

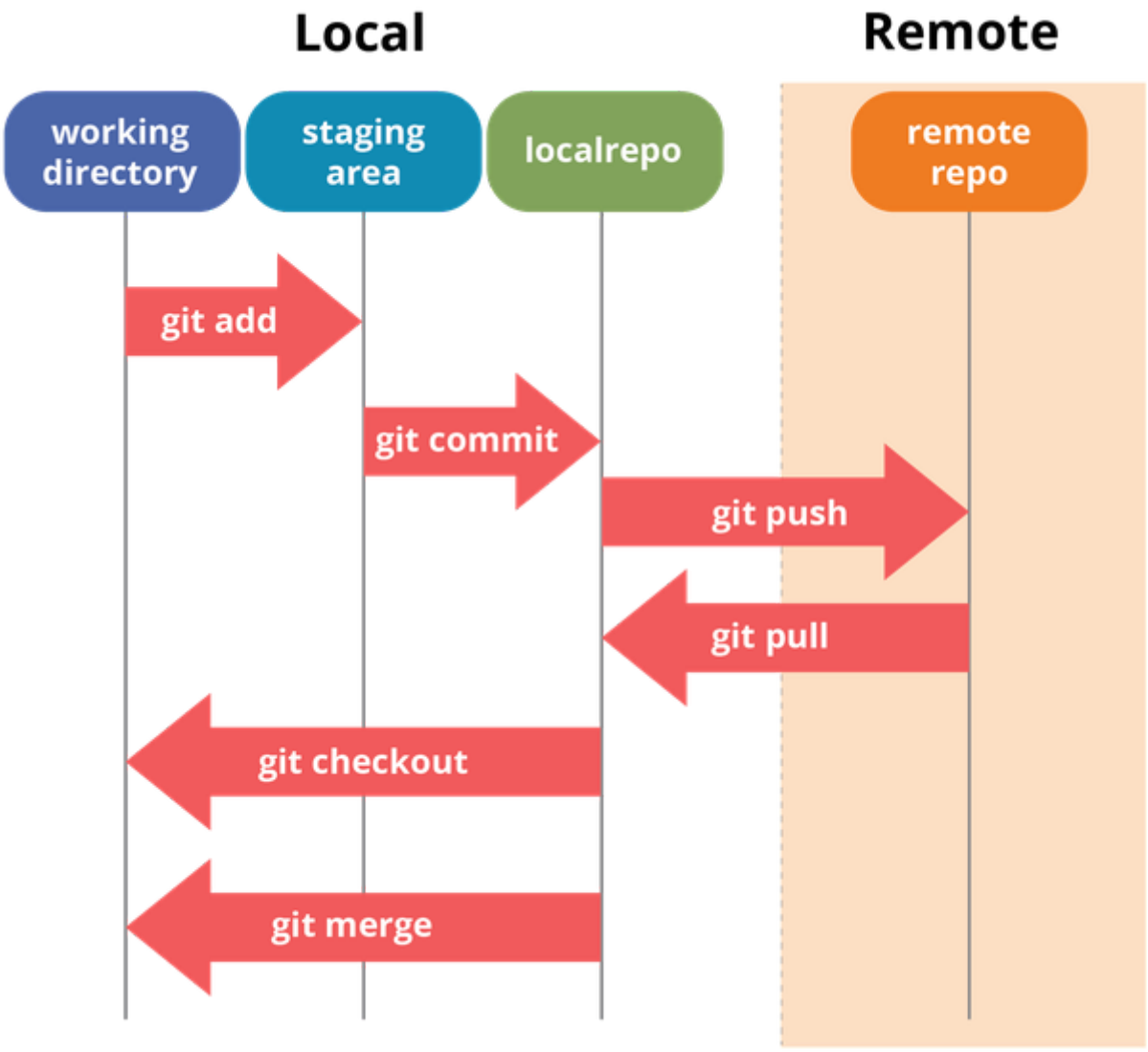
Más información: https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Instalaci%C3%B3n-de-Git

---

Antes de empezar a utilizar Git debemos ver donde queremos empezar a trabajar en nuestro sistemas de carpeta correspondiente a cada entorno de desarrollo. En nuestro caso tenemos un servidor

```
$pwd
/media/mount/4-WORKSPACE/GIT/
$mkdir proyecto1
$cd proyecto1
$touch vacio.txt
l
```

Ciclo de trabajo



Dentro de las ramas comunes que se ven en la industria, nombraremos algunas importantes:

1. Master (por defecto)
2. Development (desarrollo)
3. HotFix/BugFixing (cambios en calientes, que deben salir lo antes posibles)

Sin tener en cuenta todos los conflitos que se puedan generar en un posible merge.

Antes de iniciar con los comandos básicos cabe recalcar que necesitamos definir algunas variables globales para poder hacer commits en nuestros ficheros.

git config --list //ver las configuraciones de nuestro git
it config --list --show-origin //ver donde están las configuraciones guardadas
it config --global user.name "X"
it config --global user.email "Y"

Comandos básicos

Comando	Descripcion
git init	Inicializado repositorio Git vacío en /media/mount/4-WORKSPACE/GIT/proyecto1/.git/
ls -al	drwxrwxr-x 3 mumito mumito 4096 sep 31 21:01 . drwxrwxrwx 9 mumito mumito 4096 sep 31 21:01 .. drwxrwxr-x 7 mumito mumito 4096 sep 31 21:01 .git
git status	vemos el estado de nuestros ficheros.
git add fichero.txt / git add .	Añadimos a la base de datos de git / añadimos todo en la carpeta
git rm fichero.txt	Lo eliminamos de la base de datos de git
git show fichero.txt	Vemos todos los cambios realizados en el archivo
1) git log fichero.txt	Vemos todas las commit realizados
2) git diff b5ae3d9710c66a55b0a734e601ea94d44767b9cb 4c2723219221137bad1c01a609c834f45af6f12c	Si ponemos una version más reciente y otra más antigua, aquellos cambios salen reflejados en rojo
1) git log fichero.txt	
2.1) git reset b5ae3d9710c66a55b0a734e601ea94d44767b9cb --hard	volvemos a la version anterior eliminando todo los cambios hasta ese punto. (perdemos los cambios de staging area (git add .))
2.2)git reset b5ae3d9710c66a55b0a734e601ea94d44767b9cb --soft	volvemos atras, pero no perdemos lo que tengamos en staging area (git add .)
git diff	Vemos lo que está en el directorio, y lo que está en el staging areagit
git log --stat	Vemos el número de bytes que fueron modificados
1)git checkup [git log] fichero.txt	Nos deja ir, mirar, pasear y volver en el tiempo. algo que git reset no.
2)	
git reset HEAD	Saca archivos del area de staging. No para borrarlos, sino para que los últimos cambios de estos archivos no se envíen al último commit.
git	

Configuracion de globales de Git

Nos posicionamos en el Home.

Comandos	Descripción
ssh-keygen -t rsa -b 4096 -C "jcandela@ucm.es"s	Creamos una clave privada para conectarnos a GitHub
eval $(ssh-agent -s)	Vemos si esta corriendo el proceso ssh, por lo que mostrará el proceso que está corriendo
ssh-add ~/.shh/id_rsa	Añadimos la clave privada

Despues de crear la clave pública y privada, debemos ir a GitHub y añadimos la clave pública. Posteriormente metemos nuestra contraseña para guardar los cambios.

```
#Volvemos a la carpeta donde tenemos nuestro proyecto
git remote -v (comprobamos que tenemos la anterior url donde hemos añadido el
repositorio remoto)
#Cambiamos la url para conectarnos a git
git remote set-url origin git@github.com:HackingSecurity/pruebasGitHub.git
```

Branch

Copia exacta de lo que tengo pero en otro directorio de trabajo.

Comando	Descripción
git branch nuevaRama	Creamos un nueva rama de trabajo
git checkout master	Nos vemos a la rama de master

Merge

Antes de hacer un marge debemos situarnos en la rama donde queremos traernos los cambios de los últimos commits a la rama en la que estamos, normalmente master.

Comando	Descripción
git branch	No dice en que rama estamos situados

Comando	Descripción
git checkout master	Nos situamos en la rama donde vamos hacer el merge
git merge development	Realizamos un merge de la rama Devolopment a Master. No olvidemos que un merge es un commit
Solucionar conflictos	
git status	Vemos el estatus una vez elegido los cambios del HEAD (Rama donde estamos) y la rama de donde queremos hacer el merge.
git commit -am "Solucion de los conflictos"	Con esto se solucionaría los conflicto que se han ocasionado al modificarr los mismo ficheros y en las misma lineas.

GitHub

Una vez creada una cuenta en https://github.com/, tenemos la capacidad de crear un nuevo repositorio donde incluiremos el codigo que tenemos en nuestro directorio local o si ya lo tenemos en GitHUb clonarlo y traerlo a nuestro entorno de desarrollo.

Comandos	Descripción
git remote add origin https://github.com/HackingSecurity/pruebasGitHub.git	Añadimos la ruta a nuestro repositorio en GitHub
git remote -v	Nos muestra que tenemos un fetch (que es para traernos cosas) y push (es para enviar cosas)
git push origin master	Enviar al origen la rama master.
git pull origin master	Me traigo del repositorio los cambios **Siempre lo hacemos antes de hacer un push**, al repositorio remoto Normalmente se usa despues de un commit y queremos subir lo que tenemos al repositorio.
git pull origin master --allow-unrelated-histories	Fusionar lo que hay en el repositorio con lo que tenemos en local.ls
git config -l	Vemos todas las variables que estamos usando.