**Implantación de Git para el control de versiones en el desarrollo del software**

|  |  |
| --- | --- |
| **Git** | Es un software de control de versiones desarrollado por Linus Torvalds para aumentar la eficiencia en el desarrollo de proyectos. |
| **Flujo de trabajo** | **Directorio de trabajo**: contiene los ficheros del proyecto.  **Index**: zona intermedia donde se van almacenando cambios.  **Head**: apunta al último cambio aceptado (commit).  Tenemos una rama de desarrollo principal: **Master.** |
| **Comandos fundamentales en Git** | |
| **Git init** | Permite crear un repositorio basado en un directorio local de nuestro equipo.  Primero accedemos a nuestro directorio y posteriormente lanzamos el comando:  git init |
| **Borrar un repositorio** | Para borrar un repositorio simplemente hay que eliminar el directorio que lo contiene. |
| **Git add** | Permite registrar cambios en nuestro proyecto. Estos cambios pasarán a Index.  Ejemplo:  git add . |
| **Establecer usuario y correo electrónico** | Para establecer un usuario global para Git:  git config –global user.name “Salva”  Para establecer el correo electrónico del usuario global:  Git config –global user.email salvador.gonsan@educa.jcyl.es  git config –list (mostrar los datos asociados a la configuración de git) |
| **Git commit** | Permite aceptar los cambios añadidos para que pasen a Head.  Ejemplo:  git commit -m “Versión 1.0” |
| **Git checkout -b “rama1”** | Permite crear una rama de desarrollo con el nombre que deseemos  Ejemplo:  git checkout -b “rama1” |
| **Git checkout “rama1”** | Permite cambiar a la rama que deseemos. Tenemos que especificar su nombre como parámetro del comando.  Ejemplo:  git checkout master |
| **Git branch -d “rama1”** | Permite borrar una rama de desarrollo.  Ejemplo:  git branch -d “rama1” |
| **Git branch -a** | Muestra todas las ramas locales y remotas.  Ejemplo:  git branch -a  Con el \* marca la activa.  Muestra todas las ramas locales:  git branch |
| **Git status** | Muestra la rama activa y el estado de los cambios. |
| **Git config** | Para obtener de información sobre la configuración de git.  Ejemplo:  git config --list |
| **Conectar Git con nuestra cuenta GitHub** | **Conexión mediante SSH (protocolo que permite el acceso a un servidor remoto mediante un canal seguro en el que toda la información está cifrada. Utiliza claves para comunicar a emisor y receptor)**   1. Primero debemos crearnos una cuenta en GitHub: <https://github.com/> 2. Es fundamental crear un repositorio en GitHub      1. Posteriormente debemos crear una clave SSH para poder conectar Git con Github:   ssh-keygen -t rsa -b 4096 -C "micorreo@dominio.com"  Lo ideal es establecer una contraseña para añadir seguridad. La clave se guardará en un fichero con el nombre que deseemos, por ejemplo, **clavegithub**. El comando que anteriormente hemos escrito nos pedirá estos pasos.   1. Ahora debemos añadir nuestra clave SSH al ssh-agent.   Primero debemos asegurarnos de que el ssh-agent se encuentra en ejecución.  eval $(ssh-agent)   1. Ahora tenemos que añadir al ssh-agent la clave que guardamos anteriormente en el fichero   ssh-add clavegithub   1. Aún nos quedan más pasos. Debemos añadir la clave SSH a nuestra cuenta de GitHub          1. Hay que probar la conectividad con GitHub   ssh -T git@github.com   1. Posteriormente deberíamos iniciar un nuevo repositorio en un directorio local:   git init   1. Añadir un fichero o directorio al repositorio:   git add nombreficherodirectorio  Advertencia: puede que nos arroje warnings con alertas sobre cambios en los saltos de línea.  **Saltos de línea en Windows – CR**  **Saltos de línea en Linux – LF**  Para solucionar esto (sin que nos salten warnings): git config core.autocrlf true   1. Hacer un commit para aceptar los cambios:   git commit -m “Versión 1.0”   1. Conectamos el respositorio local con el que tenemos en GitHub (podemos obtener su dirección desde Github.com):   git remote add origenpersonalizado direccionsshrepositoriogithub  Ejemplo de dirección SSH:     1. Enviar los cambios a GitHub (enviamos la rama master a GitHub)   git push origenpersonalizado master (o el nombre de la rama que queramos)  Si nos encontramos algún error al hacer push, tal vez tengamos que resincronizar el repositorio remoto y reconstruir el árbol (aparentemente realiza de nuevo todos los commit). Para ello lanzamos estos dos comandos:  git pull –rebase repostorioremoto master  git push repositorioremoto master   1. Ya podemos consultar los cambios en github.com   **Conexión a GitHub mediante HTTPS (se trata de un protocolo de transferencia segura. Es la versión segura de HTTP y se basa en certificados que aseguran la confianza y que son vendidos y expedidos por autoridades de certificación)**  En este caso no es necesario generar una clave SSH. En cuanto hagamos el commit, ya estaremos listos para hacer un push y subir los cambios al servidor remoto.  git push direccionhttpsrepositorio master  Master es la rama que queremos subir.  Ejemplo de dirección https: |
| **git push –all origin** | Permite subir todos los commits de todas las ramas a GitHub (servidor).  Ejemplo:  Git push –all origin |
| **Clonar repositorios de GitHub** | Permite clonar repositorios existentes en nuestra cuenta de GitHub.  Ejemplo:  git clone direccionRepositorio |
| **Clonar una rama específica** | Permite clonar una rama específica del respositorio remoto.  Ejemplo: git clone—branch nombrerama direccionRepositorio |
| **pull** | Este comando permite traernos los cambios del servidor de una rama determinada y actualizar el repositorio local.  git pull direccionRepositorio master  Recomendaciones: normalmente hay que hacer antes pull que push para traernos los cambios del servidor.  Pull hace lo mismo que fetch pero fusiona el repositorio remoto con la rama master local. |
| **pull y push** | A esto podemos llamarlo **sincronizar**. Equivale a traer los cambios últimos que se hayan realizado en nuestra rama y subir los nuestros.  git pull direccionRepositorio master  git push direccionRepositorio master |
| **Resolver conflictos** | A veces queremos traer una rama del servidor, pero tiene conflictos. Se puede solucionar de la siguiente forma:  git commit -m "fixing merge conflicts" |
| **Fusionar ramas** | Es posible fusionar ramas (intenta combinar el código de ambas ramas) con el siguiente comando:  git merge rama (mezcla la rama actual con “rama”)  Por ejemplo, si estamos en la rama master, intenta combinar la rama master con la rama “rama” |
| **Resumen de conceptos Git** | Ver las imágenes de origen |
| **Chuleta externa de comandos** | <https://elbauldelprogramador.com/mini-tutorial-y-chuleta-de-comandos-git/#recomendaciones> |

|  |  |
| --- | --- |
| **Visual Studio Code - Comandos interesantes relacionados con Git** | |
| **Cambiar la terminal por defecto en Visual Studio Code** | Debemos hacer clic en View – Command Palette para abrir la paleta de comandos. Posteriormente escribiremos la siguiente orden:  Terminal: Select Default Shell.  Aquí podremos elegir:   * PowerShell * Git Bash * Command Prompt * Etc   Desde Bash podemos cambiar a PowerShell escribiendo la orden:  powershell  Desde PowerShell podemos cambiar a Bash escribiendo la orden:  bash |
| **Utilización de comandos Git desde la Paleta de Comandos** | Es posible utilizar comandos Git desde la paleta de comandos: View – Command Palette  Ejemplos:  Git: Clone – Para clonar un repositorio (dirección https).  Git: Push To – Para enviar cambios a un repositorio remoto |
| **Integración de Git en Visual Studio Code** | Es posible gestionar Git desde la interfaz gráfica de Visual Studio Code. Simplemente debemos ir al apartado adecuado.  Existe un apartado muy importante – Changes: aquí se van almacenando los cambios que vamos realizando sobre el proyecto.  También tenemos otro apartado de Commit para aceptar los cambios llevados a cabo. |