Configurar **Visual Studio Code** en español:

Ctrl+Shift+P, en el buscador escribimos “ Configure display Languaje”

Elegimos “Install addicional languages…”

Buscar y seleccionar “Spanish Language Pack…” y en la ventana de la derecha pinchar “install”. Luego reiniciar Visual Studio Code. Para desisntalar hacer lo mismo, pero pinchar en”Desinstalar”.

**Configurar Visual Studio Code** para poder trabajar en la **terminal con Git**.

Files, Preference, Settings Archivo, Preferencias, Configuración

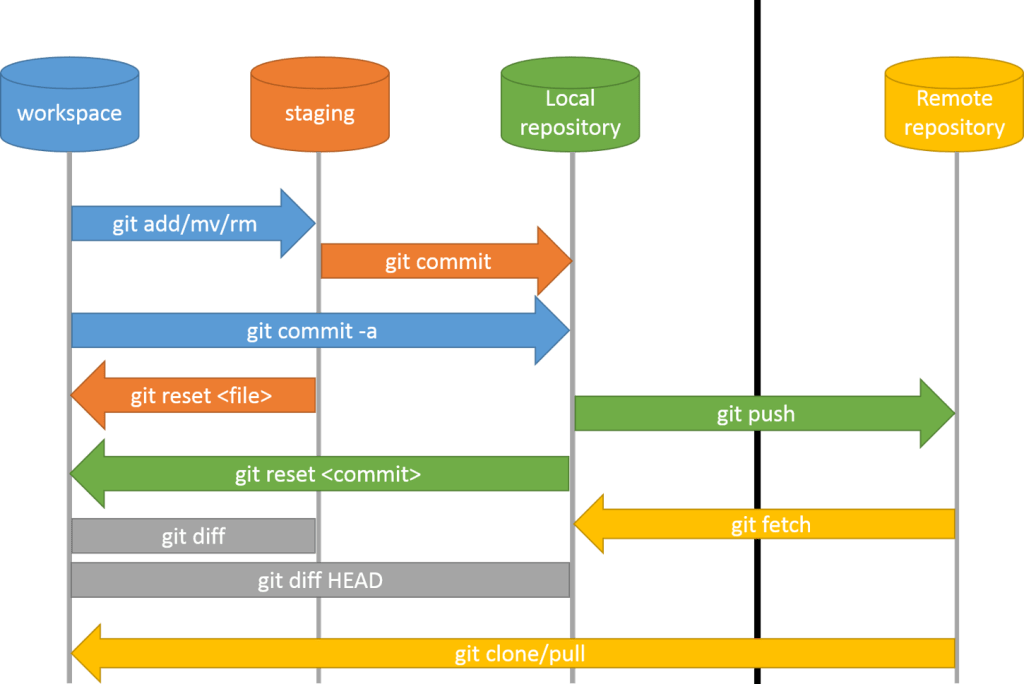
Features, Terminal Características, Terminal

En el buscador, escribir “shell”

En Terminal>Integrated>Shell>Windows, pinchar en Editar en settings.json

Añadir la ruta de la consola Git Bash y guardarla

"terminal.integrated.shell.windows": "C:\\Program Files\\Git\\bin\\bash.exe"

**Área1 Área2 Área3 github.com**

**Working Directory**

Empezar con **Git**: Hay que configurarlo y poner el nombre y el email del usuario que lo va a usar.

**git config --global user.name *"Pepe"* git config --global user.email *pepe12@gmail.com***

**git config --global --replace-all user.name *"Pepe"*****git config --global --replace-all user.email *pepe12@gmail***

**cat ~/.gitconfig** o **git config --global --list** Muestra nombre y email de quien está usando Git.

**git init** Crea repositorio Git en una carpeta (aparecerá .git oculto dentro de la carpeta) workspace.

**git status** Muestra información del repositorio y los archivos con o sin seguimiento que hay dentro.

**git status --s** Muestra información de los archivos.

**??** sin seguimiento. Área1

**A** con seguimiento. Área2

**M** sin seguimiento y modificado. Área1

**git add *nombre.ext*** Agrega el archivo especificado desde el área 1 al área 2.

**git add .** Agrega todos los archivos para hacerle seguimiento del área 1 al área 2.

**git reset *nombre.ext*** Pasa archivo del área 2 al área 1.

**git reset .** Pasa todos los archivos del área 2 al área 1.

**git rm 01.txt** Borra fichero y lo pasa al área 2. **rm 01.txt** borra fichero y lo pasa al área1.

**git checkout HEAD -- .** Recupera ficheros borrados o modificaciones que están en el área 1 o 2.

**git commit -m *"escribimos un mensaje"*** Pasa los archivos del área 2 al área 3, al Local Repository.

Graba todo lo que hayamos pasado y le da una referencia.

**git log** Ver información del commit donde nos encontramos y de los anteriores.

**git log --oneline** Muestra commit donde estamos y anteriores de forma reducida.

**git log -n 3 --oneline** Commit donde estamos y los anteriores indicados de forma reducida.

**git log --pretty=oneline** Commit anteriores y su identificación completa de forma reducida.

**git log --pretty="%h %an %ar %s"** Podemos configurar la infomación que muestra.

h -> hashid an -> autor name ar -> de cuando es el cambio s -> nombre de la revisión

**git log -p** Muestra diferencias de todos los commit anteriores.

**git log -p -n 2** Muestra diferencias del número de commit que indiquemos desde donde estamos.

**git reset --hard *6adl041*** Recuperar commit a nuestro área local. Será el nuevo master.

**git reset --hard HEAD**Recuperar el commit donde estoy, eliminando todos los cambios.

**git reset --hard HEAD~1**Recuperar versión del commit anterior. Será el nuevo master.

**git reflog** Ver historial de todos los commit y reset--hard que hemos hecho.

**git diff** Cuando hagamos git status y veamos que hay archivos modificados en el área1.

**git diff –staged o --cached** Cuando hagamos git status y veamos que hay archivos modificados en el área2.

Ver diferencias de los ficheros que hemos modificado y los que antes han sido guardados con un commit.

**git diff HEAD HEAD~2** Diferencia entre el commit donde estamos situados y 2 commit anteriores.

**git difftool** Muestra lo mismo que git diff. Pone en funcionamiento a **vimdiff**.

Si anteriormente copiamos el fichero tkdiff en C:\Program Files\Git\user\bin hace que funcione **tkdiff**.

**git blame Programa3.java** Quién ha hecho modificaciones en las líneas de un fichero.

**git blame –L 3,6 Programa3.java** Quién ha hecho modificaciones en líneas especificadas de un fichero.

**git branch** Ver las ramas que tenemos.

**git branch -a** Ver las ramas que tenemos y las remotas.

**git branch -a -v** Ver el último commit de las ramas que tenemos y de las remotas.

**git branch** ***rama1*** Crear copia de rama actual a otra rama. Debe existir commit en la rama actual.

**git branch -d *rama1*** Eliminar rama.

**git checkout *rama1*** Movernos a la que especifiquemos, a su commit más reciente.

**git checkout *master*** Ir a la rama master o ir al commit más reciente de la rama master.

**git checkout *6adl041*** Movernos al commit que especifiquemos.

**git checkout -b *rama2*** Crear y posicionarnos en una nueva rama.

**git checkout a4c5585 01.txt** Recuperar al commit en el que estemos un fichero de otro commit.

**git checkout head~2 01.txt** Recuperar al commit en el que estemos un fichero de otro commit.

**git merge *rama1*** Si estamos en master, fusiona los cambios que detecte en la rama1.

**git merge *rama1 master*** Si estamos en master, fusiona los cambios que detecte en la rama1.

**git merge *master rama1*** Si estamos en master, fusiona los cambios que detecte en la rama1.

**git tag** Muestra etiquetas de los commit.

**git tag -a java2 -m "versión 2"** Etiqueta el commit donde estemos. git checkout java2 para ir a ese commit.

**git show -a java2** Muestra información del commit que representa esa etiqueta.

**git show *6adl041*** Muestra información del commit especificado.

**GitHub:**

**Problemas con Visual Studio Code:**

-Entrar en **credenciales** de Windows y borrar las de vscodevscode.github-authenticacion/github.auth

-Entrar en **credenciales** de windows y borrar las de git:https://github.com

-Volver a pedir contraseña nueva en github.com; nos mandará un email a nuestro correo personal para que podamos resetearla. Una vez tengamos nueva contraseña, cuando hagamos un “push” desde Visual Studio Code, éste nos avisará si queremos conectar automáticamente con gitpush.com.

Creamos en github.com un repositorio (ej: Proyecto1).

------------------------------------------------------

**git remote --v** Vemos los remotos actuales.

**git remote rm origin** Elimina el remoto especificado.

------------------------------------------------------

Para subir archivos a la nube.

**git remote add origin** <https://github.com/pablosegoviasanchez/Proyecto1.git> se vincula con origin.

Decimos que en adelante se graben nuestros archivos locales en la nube en un determinado repositorio.

**git push origin master** (por ramas) **sube** la rama master a GitHub.

**git push origin rama1 sube** la rama rama1 a GitHub.

Sube los commit que hayamos creado a la nube. Refrescar la página de GitHub para comprobarlo.

En adelante bastará con escribir **git push**.

------------------------------------------------------

Para recuperar en local archivos de la nube en un directorio vacío. Crear repositorio: **git init**

**git remote add origin** <https://github.com/pablosegoviasanchez/Proyecto1.git> vincula con origin

**git pull origin** Recupera todas las ramas del repositorio remoto.

Una vez recuperada la rama o ramas que nos interesan con **git pull**, no las vemos:

**git checkout master** Ver el contenido de la rama recuperada master.

**git checkout rama1** Ver el contenido de la rama recuperada rama1.

**git pull origin rama1** Recupera una rama del repositorio remoto a rama donde estemos.

**git fetch origin** Recupera todas las ramas del repositorio remoto.

**git fetch origin rama1** Recupera una rama del repositorio remoto (master, rama1…).

Una vez recuperada la rama o ramas que nos interesan con **git feach**, no las vemos:

**git checkout master** Ver el contenido de la rama recuperada master.

**git checkout rama1** Ver el contenido de la rama recuperada rama1.

------------------------------------------------------

Para recuperar en local archivos de la nube que hayan sido modificados y subidos por otro usuario.

**git remote add origin** <https://github.com/pablosegoviasanchez/Proyecto1.git> vincula con origin

**git pull origin** Recupera modificaciones en la rama en la que estamos del repositorio.

**git pull origin rama1** Recupera modificaciones en la rama en la que estamos de la rama indicada del repositorio.

**git fetch origin** Recupera modificaciones comparando la rama en la que estamos con la rama del repositorio remoto (la oculta en rama FETCH\_HEAD)

**git merge FETCH\_HEAD** Fusiona la rama oculta con la rama en la que estamos.

**.gitignore:** hay que crear este archivo antes de hacer el **git init**.

-Crear directorio con ficheros, etc…

-Crear el fichero gitignore.txt dentro del directorio donde vayamos a crear el repositorio.

-Añadir en fichero gitignore.txt lo que debe ignorar Git. (ejemplo: \*.txt, Prog5.java, 01.html, directorio…).

-Abrir consola ***cmd.exe***

-Ir con la consola cmd a la carpeta donde queramos crear repositorio (donde está gitignore.txt).

-Con la consola, cambiar nombre al fichero que hemos creado: **ren gitignore.txt .gitignore**

-Cerrar consola cmd.

-Escribir **git init** y git status. Comprobamos que lo que hemos especificado en .gitignore es ignorado por git.

**Consola Git Bash y cmd:**

**ls -la** ver el contenido del directorio en el que estamos

**mkdir nombre** crear directorio

**rmdir nombre** borrar directorio

**cd nombre** ir a un directorio

**cd ..** retroceder de directorio

**pwd** ruta completa del directorio en el que estamos

**echo “Primera línea del fichero”>datos.txt** crear un fichero con datos

**rm datos.txt** Borrar un fichero

**git mv 7.txt 9.txt** cambiar nombre de fichero, debe estar en el área 2 o que se haya hecho commit

**ren 1.dat 3.dat** cambiar nombre de fichero. En la consola **cmd**.