



PLATAFORMA GITHUB Y GIT

Ignacio Núñez Gómez

3ºB GITT



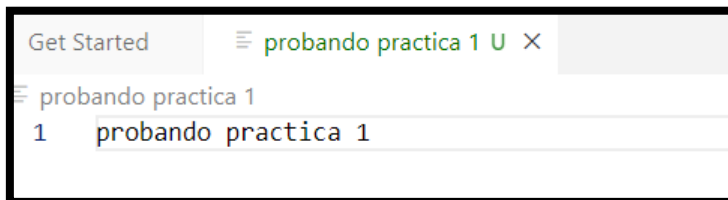
Probando comandos

1) Git clone

```
gitpod /workspace/hello-world $ git clone https://github.com/ignacionunezG/hello-world
Cloning into 'hello-world'...
remote: Enumerating objects: 38, done.
remote: Counting objects: 100% (38/38), done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 38 (delta 1), reused 31 (delta 0), pack-reused 0
Receiving objects: 100% (38/38), 58.97 KiB | 6.55 MiB/s, done.
Resolving deltas: 100% (1/1), done.
```

Git clone se utiliza para copiar un repositorio existente en mi directorio local. En este caso, estoy copiando el repositorio de 'hello-world' en mi directorio de gitpod.

2) Git status



The screenshot shows a file explorer window with a tab titled 'probando practica 1 U X'. The file list contains one item: '1 probando practica 1'.

El comando 'git status' nos dirá el estado del directorio. Nos indicará si se han producido cambios, y en qué ficheros se han producido dichos cambios. Para probar el código, primero me he creado un archivo llamado 'probando practica 1'. Al introducir el comando 'git status', se nos indica que se han producido cambios precisamente en el archivo 'probando practica 1', tal y como se puede ver en la siguiente captura de pantalla.

```
gitpod /workspace/hello-world $ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    hello-world/
    probando practica 1

nothing added to commit but untracked files present (use "git add" to track)
```

3) Git add

```
gitpod /workspace/hello-world $ git add 'probando practica 1'
```



The screenshot shows a file explorer window with a tab titled 'probando practica 1'. The file list contains one item: 'probando practica 1' with a green 'A' icon next to it, indicating it has been added to the staging area.

El comando 'git add' sirve para añadir un cambio realizado en mi directorio al 'staging area' de mi repositorio. Posteriormente se incluirán las actualizaciones realizadas en la próxima confirmación (commit).

4) Git commit

```
gitpod /workspace/hello-world $ git commit -m 'probando cosas'
[main ef24046] probando cosas
1 file changed, 1 insertion(+)
create mode 100644 probando practica 1
```

'Git commit' sirve para confirmar los cambios y actualizaciones realizadas en el directorio. Guarda los cambios hechos en el staging área, junto con una descripción de las actualizaciones realizadas. Este commit tendrá la descripción 'probando cosas', y únicamente está guardando el archivo 'probando practica 1'.

5) Git push

```
gitpod /workspace/hello-world $ git push
remote: Permission to ignacionunezG/hello-world.git denied to ignacionunezG.
fatal: unable to access 'https://github.com/ignacionunezG/hello-world.git/': The requested URL returned error: 403
gitpod /workspace/hello-world $ git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 623 bytes | 623.00 KiB/s, done.
Total 6 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To https://github.com/ignacionunezG/hello-world.git
48fe276..c4cc1f6 main -> main
```



Git push añade los cambios realizados al directorio en github. No obstante, para poder añadir los cambios primero había que actualizar los permisos de gitpod, como se puede ver en la captura de pantalla superior.

6) Git checkout

'Git checkout' sirve para moverse entre distintas ramas del repositorio. Para ilustrar lo que hace, he realizado un ejemplo.

En el archivo 'probando practica 1', he añadido la frase 'segundo commit'. He añadido los cambios de 'probando practica 1' a mi repositorio, tras haber realizado un commit llamado 'segundo commit'. A continuación, he empleado el comando 'git log --oneline'. Este comando indica cómo se identifican los distintos commits realizados. Para volver a tener los archivos y los cambios en mi directorio que tenía tras el primer commit (y no los cambios tras el segundo commit), utilizo el comando 'git checkout'. Tras 'git checkout', meteré el código identificativo del primer commit. De esa forma, como se puede ver, vuelvo a tener en mi directorio el archivo 'probando practica 1' tal y como lo tenía después del primer commit.

En la siguiente página se puede ver todo el proceso

```

Get Started  ≡ probando practica 1 M X
≡ probando practica 1
1   probando practica 1
2   |
3   | segundo commit

```

```

gitpod /workspace/hello-world $ git add 'probando practica 1'
gitpod /workspace/hello-world $ git commit -m 'segundo commit'
[main 75ae302] segundo commit
1 file changed, 4 insertions(+), 1 deletion(-)
gitpod /workspace/hello-world $ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 299 bytes | 299.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ignacionunezG/hello-world.git
c4cc1f6..75ae302  main -> main

```

```

gitpod /workspace/hello-world $ git log --oneline
1775ffc (HEAD -> main, origin/main, origin/HEAD) tercer commit
75ae302 segundo commit
c4cc1f6 cambiando credenciales
ef24046 probando cosas
48fe276 (upstream/main) Merge pull request #1 from gitt-3-pat/feature/1
5b68377 Primera iteracion
5038239 Initial commit
gitpod /workspace/hello-world $ git checkout ef24046
warning: unable to rmdir 'hello-world': Directory not empty
Note: switching to 'ef24046'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this

```

```

Get Started  ≡ probando practica 1 X
≡ probando practica 1
1   probando practica 1

```

7) Git rm

‘Git rm’ es un comando que he utilizado para eliminar un archivo creado erróneamente en mi repositorio durante la instalación de Chocolatey. Posteriormente hay que hacer add, commit y push para eliminar del repositorio el archivo ‘ChocolateInstall.ps1’

```
PS C:\Users\ignac\OneDrive\Documentos\3º\PAT\practica1\hello-world> git rm ChocolateInstall.ps1
rm 'ChocolateInstall.ps1'
```

Descargas

1)Maven

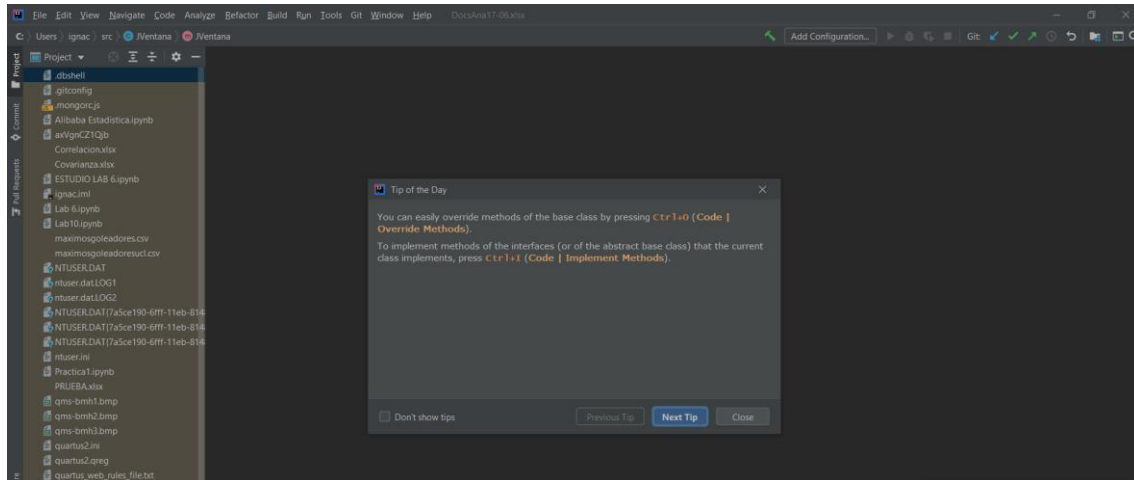
.github		14/11/2021 9:12	Carpeta de archivos	
apache-maven		14/11/2021 9:12	Carpeta de archivos	
maven-artifact		14/11/2021 9:12	Carpeta de archivos	
maven-builder-support		14/11/2021 9:12	Carpeta de archivos	
maven-compiler		14/11/2021 9:12	Carpeta de archivos	
maven-core		14/11/2021 9:12	Carpeta de archivos	
maven-embedder		14/11/2021 9:12	Carpeta de archivos	
maven-model		14/11/2021 9:12	Carpeta de archivos	
maven-model-builder		14/11/2021 9:12	Carpeta de archivos	
maven-plugin-api		14/11/2021 9:12	Carpeta de archivos	
maven-repository-metadata		14/11/2021 9:12	Carpeta de archivos	
maven-resolver-provider		14/11/2021 9:12	Carpeta de archivos	
maven-settings		14/11/2021 9:12	Carpeta de archivos	
maven-settings-builder		14/11/2021 9:12	Carpeta de archivos	
maven-slf4j-provider		14/11/2021 9:12	Carpeta de archivos	
src		14/11/2021 9:12	Carpeta de archivos	
CONTRIBUTING		14/11/2021 9:12	Archivo MD	5 KB
DEPENDENCIES		14/11/2021 9:12	Archivo	10 KB
deploySite		14/11/2021 9:12	Shell Script	1 KB
doap_Maven.rdf		14/11/2021 9:12	Archivo RDF	19 KB
Jenkinsfile		14/11/2021 9:12	Archivo	8 KB
LICENSE		14/11/2021 9:12	Archivo	12 KB
NOTICE		14/11/2021 9:12	Archivo	1 KB
pom		14/11/2021 9:12	Documento XML	27 KB
README		14/11/2021 9:12	Archivo MD	5 KB

2)Java17



Posteriormente, se añade Java17 al path del ordenador

3)IntelliJ



4)Chcolatey

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

PS C:\Users\ignac> Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; curl https://chocolatey.org/install.ps1 | iex
Installing Chocolatey on the local machine
Creating ChocolateyInstall as an environment variable (targeting 'Machine')
Setting ChocolateyInstall to 'C:\ProgramData\chocolatey'
WARNING: It's very likely you will need to close and reopen your shell
before you can use choco.
Restricting write permissions to Administrators
We are setting up the Chocolatey package repository.
The packages themselves go to 'C:\ProgramData\chocolatey\lib'
(i.e. C:\ProgramData\chocolatey\lib\yourPackageName).
A shim file for the command line goes to 'C:\ProgramData\chocolatey\bin'
and points to an executable in 'C:\ProgramData\chocolatey\lib\yourPackageName'.

Creating Chocolatey folders if they do not already exist.

WARNING: You can safely ignore errors related to missing log files when
upgrading from a version of Chocolatey less than 0.9.9.
'Batch file could not be found' is also safe to ignore.
'The system cannot find the file specified' - also safe.
chocolatey.nupkg file not installed in lib.
Attempting to locate it from bootstrapper.
APIV environment variable does not have C:\ProgramData\chocolatey\bin in it. Adding...
OVERTENCIA: Not setting tab completion: Profile file does not exist at 'C:\Users\ignac\OneDrive\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1'.
chocolatey (choco.exe) is now ready.
You can call choco from anywhere, command line or powershell by typing choco.
Run choco /? for a list of functions.
You may need to shut down and restart powershell and/or consoles
first prior to using choco.
Ensuring Chocolatey commands are on the path
Ensuring chocolatey.nupkg is in the lib folder
PS C:\WINDOWS\system32>
```