

Git y GitHub

Para configurar

Entra a Gitbash y escribe el siguiente comando

```
git config --global user.name "Martin Martinez"
```

Define el nombre de usuario

```
git config --global user.email "marnezc@gmail.com"
```

Aquí se define el email del usuario

```
$ git config --global -e
```

Aquí se muestra el archivo de configuración global del repositorio local.

```
git config --list
```

Muestra las configuraciones del git que tienes instalado

```
git init
```

Abrir Gitbash y colocarnos en la carpeta donde se encuentra nuestro proyecto luego usare el comando para inicia un repositorio local

```
ls
```

Muestra los ficheros del directorio actual

```
Pwd
```

Muestra la dirección del directorio actual en el que te encuentras.

```
ll
```

Muestra los permisos que tienen los ficheros o directorios de la carpeta actual

```
cat
```

Muestra el contenido del fichero que elijas, por ejemplo

```
$ cat index.php
```

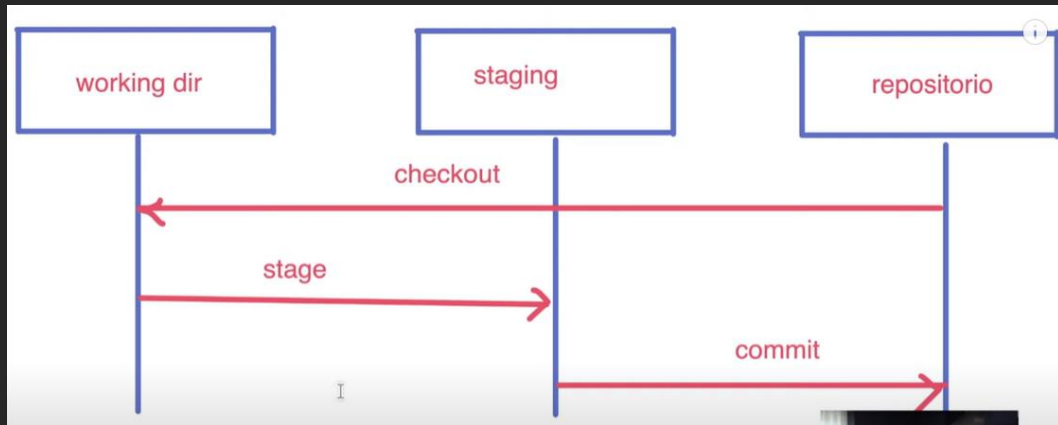
```
ls -alh
```

Muestra todos los directorios con sus respectivos permisos.

```
touch
```

Crea un fichero

```
$ touch file.txt
```



git status

Sirve para saber cual es el estatus del directorio en el que estamos trabajando.

```
marti@LAPTOP-7G5MFC2D MINGW64 ~/Documents/workspace-sts-3.9.13.RELEASE (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .metadata/
    Servers/
    dia1_ejercicio3/
    dia1_ejercicio4/
    dia1_primitivos/
    dia2_ejercicio10/
    dia2_ejercicio5/
    dia2_ejercicio6/
    dia2_ejercicio7/
    dia2_ejercicio8/
    dia2_ejercicio9/
    dia2_ejercicios/
    dia2_estructuras_control/
    dia3_ejercicio1/
    dia3_ejercicio2/
    dia3_ejercicio3/
    dia3_ejercicio4/
    dia3_ejercicio5/
    dia3_ejercicio6/

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

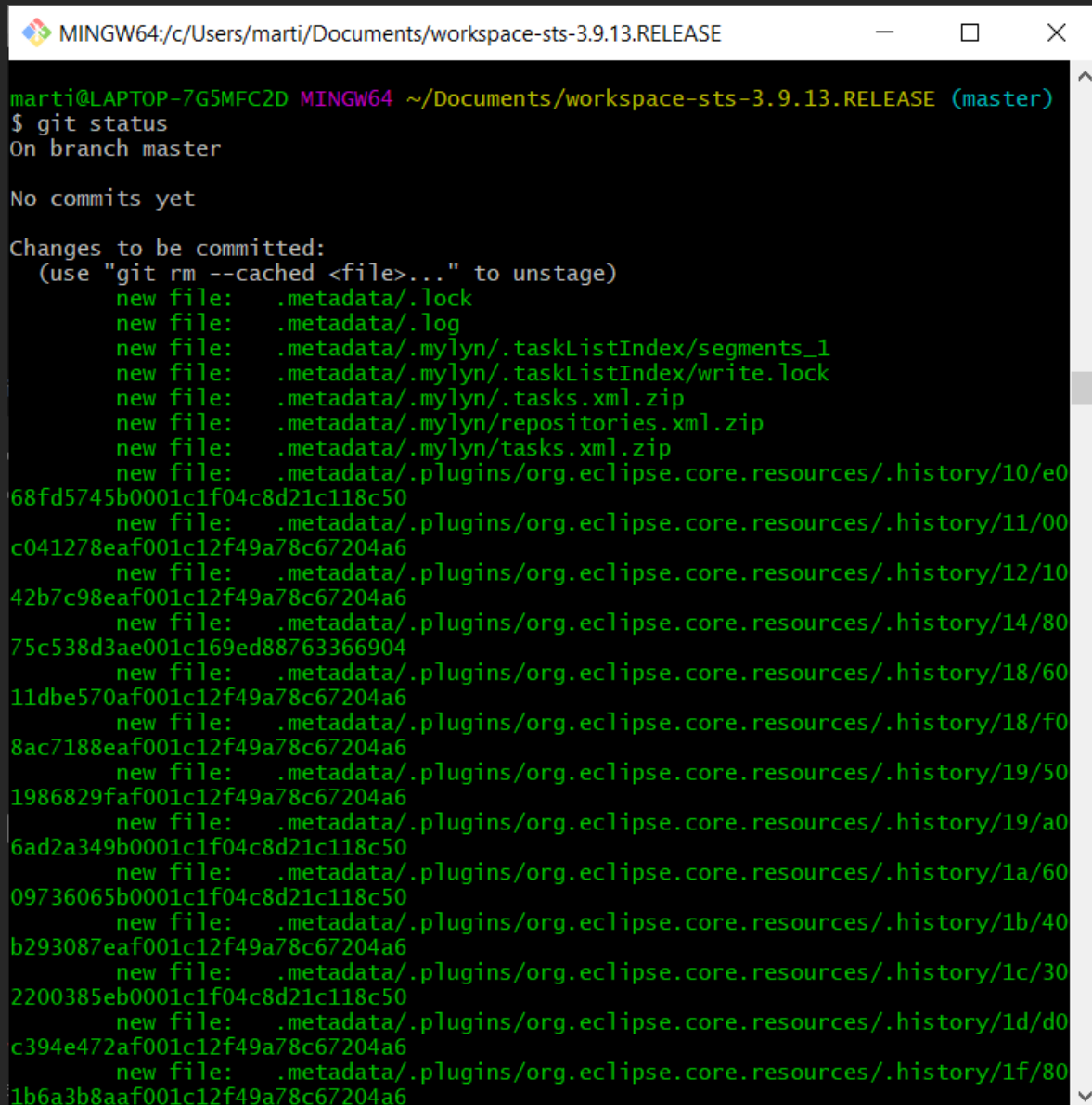
Como se puede apreciar en esta imagen al mandar el comando `git status` dice que no están listados ninguno de nuestros archivos por GIT

Git status -s

Muestra una vista mas simplificada de el estatus de los archivos en git

git add -A

Con este comando podemos listar los archivos que se encuentran en la carpeta que nos encontramos.

A screenshot of a Windows command prompt window titled "MINGW64:/c/Users/marti/Documents/workspace-sts-3.9.13.RELEASE". The prompt shows the user "marti@LAPTOP-7G5MFC2D" in the "MINGW64" environment, located at "~/Documents/workspace-sts-3.9.13.RELEASE" on the "master" branch. The command "\$ git status" has been executed. The output indicates "On branch master" and "No commits yet". It then lists "Changes to be committed:" and provides a list of new files to be staged, including metadata files and Eclipse core resources history files. The list of files is as follows:

```
new file:   .metadata/.lock
new file:   .metadata/.log
new file:   .metadata/.mylyn/.taskListIndex/segments_1
new file:   .metadata/.mylyn/.taskListIndex/write.lock
new file:   .metadata/.mylyn/.tasks.xml.zip
new file:   .metadata/.mylyn/repositories.xml.zip
new file:   .metadata/.mylyn/tasks.xml.zip
new file:   .metadata/.plugins/org.eclipse.core.resources/.history/10/e068fd5745b0001c1f04c8d21c118c50
new file:   .metadata/.plugins/org.eclipse.core.resources/.history/11/00c041278eaf001c12f49a78c67204a6
new file:   .metadata/.plugins/org.eclipse.core.resources/.history/12/1042b7c98eaf001c12f49a78c67204a6
new file:   .metadata/.plugins/org.eclipse.core.resources/.history/14/8075c538d3ae001c169ed88763366904
new file:   .metadata/.plugins/org.eclipse.core.resources/.history/18/6011dbe570af001c12f49a78c67204a6
new file:   .metadata/.plugins/org.eclipse.core.resources/.history/18/f08ac7188eaf001c12f49a78c67204a6
new file:   .metadata/.plugins/org.eclipse.core.resources/.history/19/501986829faf001c12f49a78c67204a6
new file:   .metadata/.plugins/org.eclipse.core.resources/.history/19/a06ad2a349b0001c1f04c8d21c118c50
new file:   .metadata/.plugins/org.eclipse.core.resources/.history/1a/6009736065b0001c1f04c8d21c118c50
new file:   .metadata/.plugins/org.eclipse.core.resources/.history/1b/40b293087eaf001c12f49a78c67204a6
new file:   .metadata/.plugins/org.eclipse.core.resources/.history/1c/302200385eb0001c1f04c8d21c118c50
new file:   .metadata/.plugins/org.eclipse.core.resources/.history/1d/d0c394e472af001c12f49a78c67204a6
new file:   .metadata/.plugins/org.eclipse.core.resources/.history/1f/801b6a3b8aaf001c12f49a78c67204a6
```

Al mandar un git status nuevamente podemos darnos cuenta que ahora los archivos que tenemos en la carpeta ya están listados.

git add archive.txt Esta es otra forma de indicar los cambios

```
git rm --cached .gitignore
```

Este comando ignora el fichero .gitignore

```
git commit -m "Agregando mis primeros archivos java"
```

```
git log
```

Muestra todos los commits que se han hecho en este repositorio, es decir muestra el historial de todos los commits.

```
Git log --oneline
```

Muestra todos los commits que se han hecho en orden pero solo una línea. Es decir que se puede ver una línea más simplificada.

```
git rm archivo.txt
```

Elimina el archivo (archivo.txt) pero nos ahorra el paso de agregarlo con git add y solo nos quedará el paso de commitear el cambio.

```
git restore --staged archivo.txt
```

Quita el archivo del staged para que pueda ser restaurado

```
git restore archivo1.txt
```

Restaura el archivo que habíamos pasado al stage.

```
mv archivo1.txt archivoABC.txt
```

Cambia el nombre del archivo

```
Git mv archivo.txt archivoABC.txt
```

Cambia el nombre del archivo pero aparte lo agrega al staged para después solo comitearlo.

```
git clone
```

Sirve para clonar un repositorio. Por ejemplo: `git clone http://lauri.com`

...or create a new repository on the command line

```
echo "# practicaJava" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/marnezcam/practicaJava.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/marnezcam/practicaJava.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

`git diff`

Sirve para saber cuáles son los archivos que han sido modificados

`git diff --staged`

Muestra los cambios que se acaban de realizar pero que aun se encuentran en el stage es decir aun sin comitear

Para subir el archivo a la master se debe de escribir el siguiente comando

`git push origin master`

`git pull origin master`

Descarga la última versión del código se debe de

`git remote rm origin`

Remueve un repositorio remoto.

`git remote add origin https://github.com/marnezcam/practicaJava.git`

Agrega un repositorio remoto al proyecto en el que te encuentras.

Para poder subir nuestro proyecto a GitHub primero debemos de agregar un repositorio y esto se hace con este comando que es obtenido cuando creas el repositorio en la página de GitHub

`git remote -v`

Muestra los directorios remotos de el proyecto en el que te encuentras.

`git branch martinez`

Crea una nueva rama en mi repositorio llamada "martinez"

`git checkout martinez`

Nos cambiamos a la nueva rama llamada martinez

`git branch`

Nos muestra todas las ramas que se encuentran en nuestro equipo

`git push -u origin martinez`

git push origin martinez

sube los cambios a la rama martinez con un identificador

`git branch --merged`

Muestra todas las ramas que hemos mergeado o añadido a nuestro repositorio local

`git merge martinez`

actualiza los cambios hechos en mi rama localmente hablando

`git push origin --delete martinez`

Elimina mi rama de mi repositorio remoto

`git branch -a`

Muestra todas las ramas que existen en mi repositorio local

`Git branch -d martinez`

Esto borrara mi rama "martinez" de mi repositorio local

Así se trabaja con git y github

Paso 1

Nos pasamos a la rama maestra con:

`git checkout master`

Paso 2

Hacemos un pull de la rama maestra con:

`git pull origin master`

Paso 3

Creamos una nueva rama con:

`git branch nuevarama`

Paso 4

nos cambiamos a la nueva rama con

`git checkout nuevarama`

paso 5

hacemos las modificaciones que tengamos que hacer

paso 6

Agregamos los cambios a nuestro stagin con:

```
git add -A
```

paso 7

Con esto guardamos los cambios en el repositorio local y se prepara todo para subirlo

```
git commit -m "Actualizando errores"
```

paso 8

Con esto subimos los cambios a nuestra rama remota

```
git push origin nuevarama
```

paso 9

Con este commando nos pasamos a la rama maestra

```
git checkout master
```

paso 10

Con este comando compruebo si están actualizados los cambios en la rama de mi repositorio

```
git branch --merge
```

paso 11

con este comando actualizo los cambios a la rama de mi repositorio local

```
git merge nuevarama
```

paso 12

con este comando actualizamos en nuestro repositorio remoto los cambios incluidos los de la rama creada

```
git push origin master
```