

Gestión de Repositorios y Control de Código

Comandos esenciales de Git para gestionar repositorios locales y remotos, configurar cuentas, manejar ramas y realizar operaciones comunes.

Uso de Git para Control de Versiones y Repositorios de código

- Configuración de cuenta:
 - `git config user.name "<nombre>"`: Configura el nombre de usuario para el sistema Git.
 - `git config user.email "<email>"`: Configura el email del usuario para el sistema Git.
- Iniciar / Asociar repositorio / clonar:
 - `git init`: Inicializa un nuevo repositorio de Git en el directorio actual.
 - `git remote add origin <URL>`: Asocia el repositorio local con un repositorio remoto.
 - `git clone <URL>`: Clona un repositorio remoto en el directorio actual.
- Agregar / Remover archivos y subir:
 - `git add .`: Añade todos los cambios en el directorio actual al área de preparación.
 - `git rm <archivo>`: Elimina un archivo del repositorio y del sistema de archivos.
 - `git commit -m "<mensaje>"`: Registra los cambios en el repositorio con un mensaje descriptivo.
 - `git push`: Envía los commits locales a un repositorio remoto.
 - `git pull`: Actualiza el repositorio local con los cambios del repositorio remoto.
 - `git fetch`: Obtiene actualizaciones del repositorio remoto sin fusionarlas con tu copia local.
- Manejo de ramas:
 - `git branch -a`: Muestra todos los branches disponibles, tanto locales como remotos.
 - `git checkout -b <nuevo-branch>`: Crea un nuevo branch y se cambia automáticamente a él.

- `git branch -d <branch>`: Elimina una rama local.
- `git push origin --delete <branch>`: Elimina una rama remota.
- Historial y estado:
 - `git status`: Muestra el estado de los archivos en el área de preparación y el repositorio.
 - `git diff`: Muestra la diferencia entre los archivos en el área de trabajo y el último commit.
 - `git log`: Muestra el historial de commits.
 - `git log HEAD..origin/master`: Muestra los commits que están en origin/master pero no en HEAD.
- Etiquetas y estado de commits:
 - `git tag <nombre-etiqueta>`: Crea una nueva etiqueta (tag) para el commit actual.
 - `git checkout <commit>`: Cambia el área de trabajo al estado de un commit específico.

Parte Práctica

Descargar e instalar Git para el SO de tu computadora

<https://git-scm.com/downloads>

Crear una cuenta en GitHub

<https://github.com/>

En la cuenta de GitHub se pueden crear tantos repositorios como sean necesarios.

Estos repositorios se crean PÚBLICOS si se desea compartir con otras personas y siempre deben tener un archivo README.md, así que al momento de crear el nuevo repositorio recordar marcar que se cree el archivo README.

Una vez creado el repositorio remoto, hay que clonarlo en nuestra computadora (repositorio local), para esto se puede utilizar el código HTTPS o SSH.

Código HTTPS

se utiliza para clonar un repositorio que nos es propio o que no se van a realizar modificaciones a los archivos y luego volver a subir las modificaciones a ese repositorio remoto.

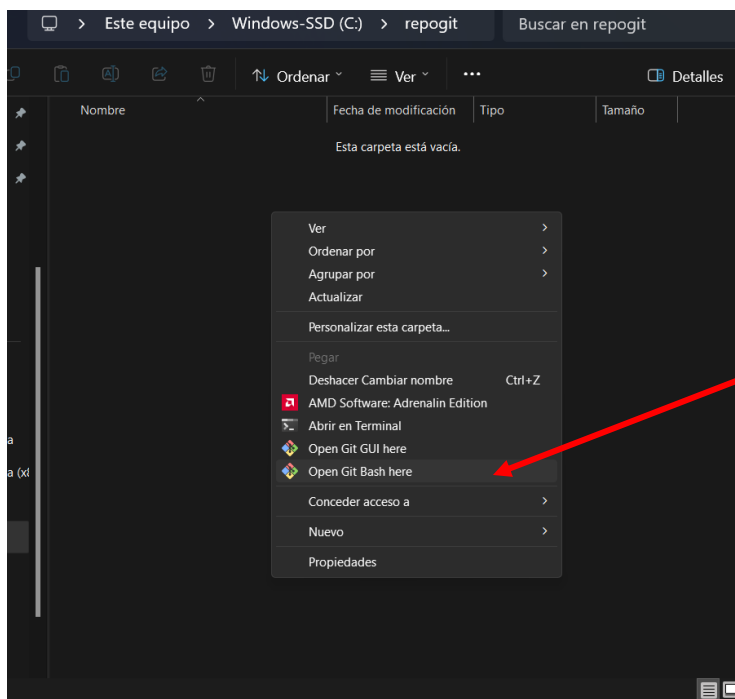
De todos modos se puede utilizar para subir modificaciones pero GitHub solicitará credenciales, usuario, contraseña o token para validar el usuario.

Código SSH

Por medio de este código el usuario se valida automáticamente, sin necesidad de validar usuario, contraseña o token cada vez que se quiere subir alguna modificación al repositorio remoto. Para esto es necesario generar un juego de llaves pública y privada en nuestra computadora y guardar la llave pública en el repositorio remoto. En el momento de la conexión se realiza de manera automática el cruzamiento de llaves que valida al usuario y permite el acceso.

Como generar las llaves pública y privada

Abrir una terminal de Git Bash (siempre conviene hacerlo en el directorio donde se vamos a clonar el repositorio remoto), con botón derecho del mouse (en cualquier lugar vacío de la ventana del administrador de archivos) seleccionar **OPEN GIT BASH HERE**



Se abrirá la una terminal de comandos de Git Bash, que muestra el usuario, nombre de la máquina y el path donde estamos ubicados, luego nos devuelve el prompt para ejecutar comandos.

```
MINGW64:/c/repogit
rosal@LENOVORBI MINGW64 /c/repogit
$
```

Generar clave pública y clave privada

ssh-keygen -t ed25519 (dar enter en todos los casos)

```
rosal@LENOVORBI MINGW64 /c/repogit
$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/c/Users/rosal/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/rosal/.ssh/id_ed25519
Your public key has been saved in /c/Users/rosal/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:fLXu5hoihNQ56zoXn0jBMbLYvsj7cojyxQIhkApbhME rosal@LENOVORBI
The key's randomart image is:
+--[ED25519 256]--+
|==...
|Eo +.+
|o+ +.B+o
|o o.B.++
|+. =O.S
|. oo*
|o +o+...
|o.o.o.o.
|.o .+o
+-----[SHA256]-----+
```

Las llaves fueron creadas en /c/Users/rosal/.ssh/ La carpeta .ssh es oculta

Se puede la lista de archivos utilizando el comando

ls -la [path]

```
rosal@LENOVORBI MINGW64 /c/repogit
$ ls -la /c/Users/rosal/.ssh/
total 15
drwxr-xr-x 1 rosal 197609 0 Nov 24 16:07 ./
drwxr-xr-x 1 rosal 197609 0 Nov 13 21:38 ../
-rw-r--r-- 1 rosal 197609 411 Nov 24 16:07 id_ed25519
-rw-r--r-- 1 rosal 197609 97 Nov 24 16:07 id_ed25519.pub
-rw-r--r-- 1 rosal 197609 92 Oct 5 23:36 known_hosts
```

← Llave privada

← Llave pública

La llave pública es la que se comparte, la privada nunca.

Ver el contenido de la llave pública

```
cat /c/users/rosal/.ssh/id_ed25519.pub
```

```
rosa1@LENOVORBI MINGW64 /c/repogit
$ cat /c/users/rosal/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGPhPwLifn9Be9XJcpJPb/C2BsIQYsgf0J9HrLUCtSU1 rosa1@LENOVORBI
```

Copiar la clave pública, ir a Git, loguearse y pegar la clave pública en el perfil del usuario

Agregar una nueva llave SSH



Confirm access



Signed in as @rinsaurralde

Password

.....

[Forgot password?](#)

Confirm

Tip: You are entering [sudo mode](#). After you've performed a sudo-protected action, you'll only be asked to re-authenticate again after a few hours of inactivity.

Settings

Type to search

You have successfully added the key 'Clave pública de VMPuebas'.



rinsaurralde (rinsaurralde)

Your personal account

[Go to your personal profile](#)

- Public profile
- Account
- Appearance
- Accessibility
- Notifications

Access

- Billing and plans
- Emails
- Password and authentication

SSH keys

[New SSH key](#)

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication keys



Clave pública de VMPuebas

SHA256:AvD4812Faq23nF8XD5I/3w88wv15CmtawP9Eshst76s

Added on Aug 31, 2024

Never used — Read/write

Delete

Check out our guide to [connecting to GitHub using SSH keys](#) or troubleshoot [common SSH problems](#).

Ahora se puede ver el código SSH para clonar el repositorio remoto

rinsaurralde / vmvagrant_repo

Type to search

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

vmvagrant_repo Public

Pin Unwatch 1 Fork 0 Star 0

main 1 Branch 0 Tags

Go to file Add file <> Code

Local Codespaces

Clone

HTTPS SSH GitHub CLI

git@github.com:rinsaurralde/vmvagrant_repo.git

Use a password-protected SSH key.

Open with GitHub Desktop

Download ZIP

About

No description, website, or topics provided.

Readme

Activity

0 stars

1 watching

0 forks

Releases

No releases published

[Create a new release](#)

Packages

Copiar el código SSH, ir a la terminal de Git Bash y clonar el repo remoto

git clone [código_ssh]

```
vagrant@VMPruebas:~$ git clone git@github.com:rinsaurrealde/vmvagrant_repo.git
Cloning into 'vmvagrant_repo'...
The authenticity of host 'github.com (20.201.28.151)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvC0qU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.
```

Cambiar al directorio del repo y crear un archivo

```
vagrant@VMPruebas:~$ cd vmvagrant_repo
vagrant@VMPruebas:~/vmvagrant_repo$ ls
README.md  archivo1.txt
vagrant@VMPruebas:~/vmvagrant_repo$ history > historial.txt
vagrant@VMPruebas:~/vmvagrant_repo$ ls
README.md  archivo1.txt  historial.txt
```

Git status, Git add .

```
vagrant@VMPruebas:~/vmvagrant_repo$ 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)
    historial.txt

nothing added to commit but untracked files present (use "git add" to track)
vagrant@VMPruebas:~/vmvagrant_repo$ git add .
vagrant@VMPruebas:~/vmvagrant_repo$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   historial.txt
```

Git commit, Git push

```
vagrant@VMPruebas:~/vmvagrant_repo$ git commit -m "commit clase2"
Author identity unknown

*** Please tell me who you are.

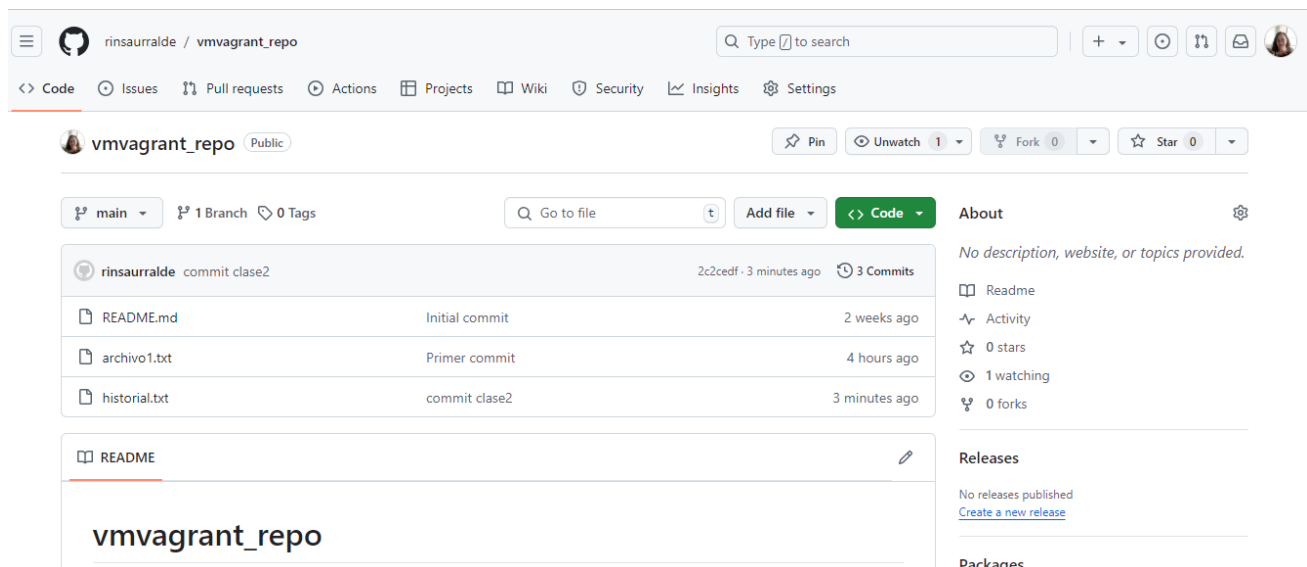
Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: empty ident name (for <vagrant@VMPruebas>) not allowed
vagrant@VMPruebas:~/vmvagrant_repo$ git config --global user.email "r.insaurralde@sistemas-utnfra.com.ar"
vagrant@VMPruebas:~/vmvagrant_repo$ git config --global user.name "rinsaurralde"
vagrant@VMPruebas:~/vmvagrant_repo$ git commit -m "commit clase2"
[main 2c2cedf] commit clase2
 1 file changed, 19 insertions(+)
 create mode 100644 historial.txt
vagrant@VMPruebas:~/vmvagrant_repo$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 549 bytes | 549.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:rinsaurralde/vmvagrant_repo.git
 74eca61..2c2cedf  main -> main
```

Ir a Git, dar F5 y verificar que se subió el archivo:



The screenshot shows the GitHub interface for the repository 'vmvagrant_repo'. The repository is public and has 1 branch (main) and 0 tags. The commit history shows a single commit 'commit clase2' by 'rinsaurralde' 3 minutes ago. The commit message is 'commit clase2'. The commit details show three files: 'README.md' (Initial commit, 2 weeks ago), 'archivo1.txt' (Primer commit, 4 hours ago), and 'historial.txt' (commit clase2, 3 minutes ago). The repository has 0 stars, 1 watching, and 0 forks. The README section is visible, showing the repository name 'vmvagrant_repo'.