Comenzamos a las 13:10

git101

Taller introductorio a control de versiones con **Git y GitHub**

Quién soy

Han Rodríguez · Rody

ITC - 6° semestre

- Investigación IA en Tec
- PE Fellow en MLH
- SWE Intern en Intel

Situación

```
[gonzalpi@fedora ~]$ g++ main.cpp -o main

[gonzalpi@fedora ~]$ ./main
El código se ejecutó con éxito
```

El código funciona.

Sigues desarrollando.

Problema

```
[gonzalpi@fedora ~]$ g++ main.cpp -o main

[gonzalpi@fedora ~]$ ./main
Segmentation fault (core dumped)
```

El código deja de funcionar,
no lo respaldaste antes
y no tienes cómo regresar.

Solución

```
int sum = 0;
// for (int i=0; i<n; i++) {
//     sum += i + 1;
// }
sum = n * (n + 1) / 2;
```

Comentar el código que funciona
y borrarlo cuando lo demás funcione.

Solución

Llevar un control de versiones.

- `actividad_1.3.txt`
- `actividad_1.3_v2.txt`
- `actividad_1.3_v3.txt`
- `actividad_1.3_final.txt`
- `actividad_1.3_final_v2.txt`

Desventaja

- Nombres irregulares
- Uso ineficiente de almacenamiento

Solución

`git` : un programa de control de versiones.

Nota: `git` no es GitHub.

¿Cómo funciona `git`?

- Creo un proyecto/repositorio de `git`.

`(0)` Creo un archivo vacío y lo registro.

```
// main.cpp
```

(1) Agrego código y lo registro.

```
// main.cpp

+ int sum = 0;
+ for (int i=0; i<n; i++) {
+     sum += i + 1;
+ }
```

(2) Elimino código y lo registro.

```
// main.cpp

   int sum = 0;
-  for (int i=0; i<n; i++) {
-      sum += i + 1;
-  }
```

(3) Agrego código más eficiente y lo registro.

```
// main.cpp

  int sum = 0;
+ sum = n * (n + 1) / 2;
```

Cada cambio está registrado

- (0) Creo un archivo vacío
- (1) Agrego código
- (2) Elimino código
- (3) Agrego código más eficiente

```
// main.cpp

  int sum = 0;
  sum = n * (n + 1) / 2;
```

Práctica 1

Crear un repositorio y añadir un archivo

Prerrequisitos

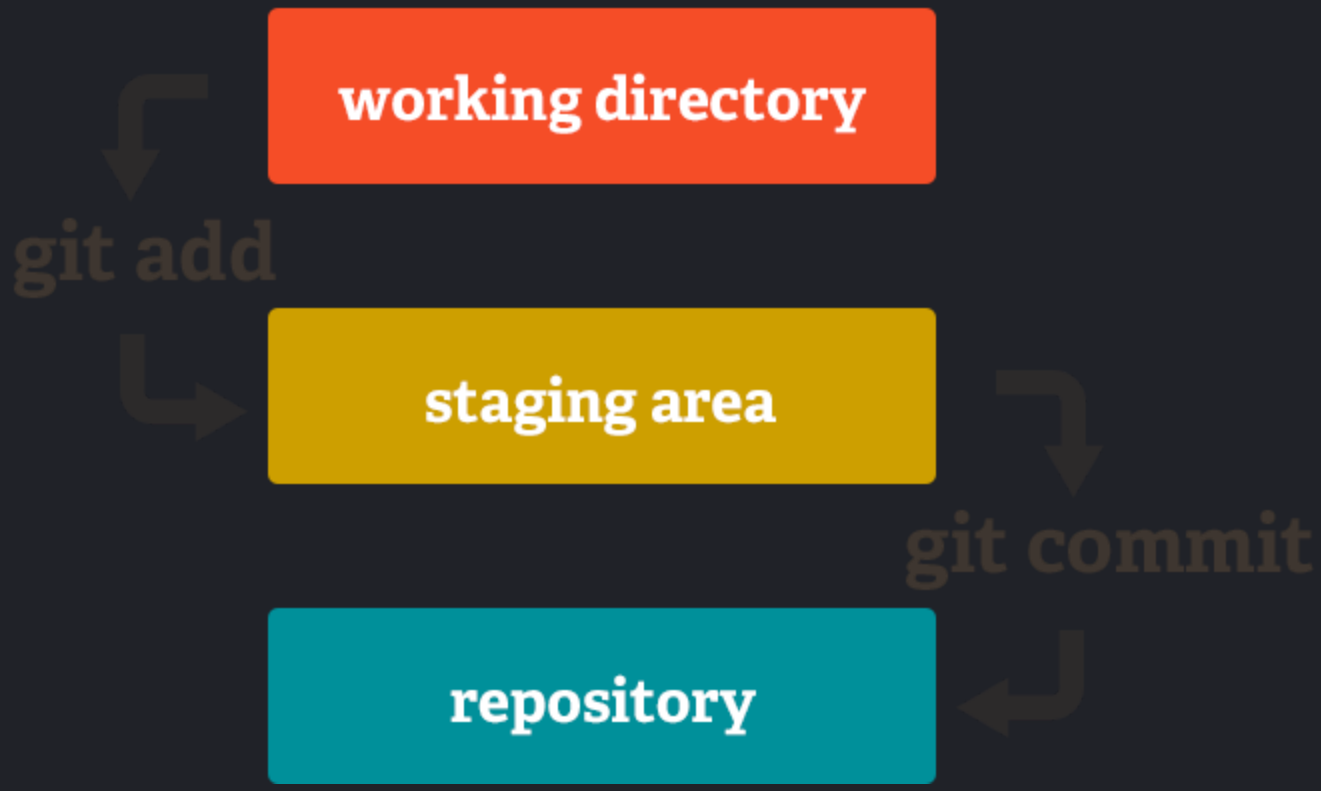
Instalar `git`

git-scm.com/download

Pasos

1. `git config --global` - configurar `git`
   - `user.name` - nombre
   - `user.email` - correo
   - `init.defaultBranch` - rama por defecto
2. `git init` - crear un repositorio
3. Crear un archivo
4. `git add` - añadir un archivo
5. `git commit` - registrar un archivo

Flujo de `git`



Lucas Maurer, 2017
https://medium.com/@lucasmaurer/git-gud-the-working-tree-staging-area-and-local-repo-a1f0f4822018

Demo con `git status` y `git log`

GitHub

GitHub es una plataforma de alojamiento de proyectos.

Podemos tener una copia de nuestro registro de `git`.

Práctica 2

Configurar GitHub y subir nuestro repositorio

Prerrequisitos

Tener una cuenta de GitHub

github.com

Pasos

1. Configurar acceso a GitHub
   - Generar llave SSH

```
ssh-keygen -t rsa -C "mail@example.com"
cat ~/.ssh/id_rsa.pub
```

   - Copiar y añadir a github.com/settings/keys
   - Probar conexión

```
ssh -T git@github.com
```

Pasos

2. Crear repositorio en github.com/new
3. Añadir dirección remota

```
git remote add origin <url>
```

4. Subir repositorio

```
git push origin main
```

5. Hacer modificaciones y repetir

Práctica 3

Hacer una contribución a un repositorio colaborativo

Pasos

1. Enviar tu usuario o correo de GitHub por el chat
2. `git clone` - clonar el repositorio
   - github.com/TECoding/2023-05-30-workshop-git101
3. `git pull` - descargar últimos cambios
4. Crear un archivo de texto con tus iniciales de nombre
5. Subir tu archivo

```
git add LARG.txt
git commit -m "Rody"
git push
# git pull y repetir si no fue posible subir tus cambios
```