

Laboratorio GIT.

Steven Gerardo Chacón Salazar

Escuela de Ingeniería Eléctrica
Programación Bajo Plataformas Abiertas
Facultad de Ingeniería Eléctrica
Universidad de Costa Rica

M.Sc. Julián Alberto Gairaud Benavides

III Ciclo 2022

1. Tome un screenshot de su editor de texto mostrando los conflictos de ambas ramas (1 punto).

```
import funcionesTarea

gato1 = funcionesTarea.gato()
<<<<<<< HEAD
gato1.nombre = "Caramelo"
=====
gato1.nombre = "Oreo"
>>>>>>> feature
gato1.color = "negro"
gato1.edad = 4
gato1.anadirlista()

gato2 = funcionesTarea.gato()
gato2.nombre = "Pelusa"
gato2.color = "cafe"
gato2.edad = 3
gato2.anadirlista()

gato3 = funcionesTarea.gato()
gato3.nombre = "Limón"
gato3.color = "blanco"
gato3.edad = 2
gato3.anadirlista()

funcionesTarea.printlista()
```

*Figura 1: Conflicto al hacer **merge** en ambas ramas **main** y **feature**.*

2. Incluya los cambios en su repositorio remoto, posteriormente corra el comando “git log --oneline --graph” y tome un screenshot. (2 puntos).

```

stevonsa@stevonsa-VirtualBox:~/Git/LaboratorioGit/gatosGit$ git rebase main
First, rewinding head to replay your work on top of it...
Fast-forwarded feature to main.
stevonsa@stevonsa-VirtualBox:~/Git/LaboratorioGit/gatosGit$ git status
On branch feature
Your branch is ahead of 'origin/feature' by 4 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
stevonsa@stevonsa-VirtualBox:~/Git/LaboratorioGit/gatosGit$ cat gatos.py
import funcionesTarea

gato1 = funcionesTarea.gato()
gato1.nombre = "Caramelo"
gato1.color = "negro"
gato1.edad = 4
gato1.anadirlista()

gato2 = funcionesTarea.gato()
gato2.nombre = "Pelusa"
gato2.color = "cafe"
gato2.edad = 3
gato2.anadirlista()

gato3 = funcionesTarea.gato()
gato3.nombre = "Limón"
gato3.color = "blanco"
gato3.edad = 2
gato3.anadirlista()

funcionesTarea.printlista()
stevonsa@stevonsa-VirtualBox:~/Git/LaboratorioGit/gatosGit$ git push
Total 0 (delta 0), reused 0 (delta 0)
To github.com:stevonsa/LaboratorioGit.git
 10ec5c1..a2ccfb3  feature -> feature

```

Figura 2: Inclusión de los cambios en el repositorio.

```

stevonsa@stevonsa-VirtualBox:~/Git/LaboratorioGit/gatosGit$ git log --oneline --graph
* a2ccfb3 (HEAD -> feature, origin/main, origin/feature, origin/main, main) Merge branch 'main' of github.com:stevonsa/LaboratorioGit into main Se cambio el nombre del gato 1 de oreo a caramelo.
* 40bfff7a Merge branch 'feature' into main
* 77999ea Se cambio el nombre del gato1 a Caramelo
* 8ef973a Se cambio el nombre del gato1 a Caramelo
* 10ec5c1 Se creó un gato nuevo gato3, con nombre limón, color blanco y de 2 años de edad.
* 6a09dd2 Se cambio el nombre del gato1 a Oreo.
* 10d0120 included gato2 and list printing
* c2b077a corrected mistakes
* d02aed2 included gato1
* 7ab0adc included functions file
* a13fd00 first commit
stevonsa@stevonsa-VirtualBox:~/Git/LaboratorioGit/gatosGit$

```

Figura 3: Utilización del comando “*git log --oneline --graph*” en git.

3. Del manual de git log que aparece al usar el comando --help, tome un screenshot de su terminal mostrando para qué sirve el comando --graph, otro mostrando qué significa el comando --oneline y otro más mostrando para qué fue “diseñado” oneline (3 screenshots en total para el punto 8) (3 puntos).

```
--graph
  Draw a text-based graphical representation of the commit history on the left hand
  side of the output. This may cause extra lines to be printed in between commits,
  in order for the graph history to be drawn properly. Cannot be combined with
  --no-walk.

  This enables parent rewriting, see History Simplification above.

  This implies the --topo-order option by default, but the --date-order option may
  also be specified.

--show-linear-break[=<barrier>]
  When --graph is not used, all history branches are flattened which can make it
  hard to see that the two consecutive commits do not belong to a linear branch.
  This option puts a barrier in between them in that case. If <barrier> is
  specified, it is the string that will be shown instead of the default one.
```

*Figura 4: funcionalidad del comando **–graph**.*

```
--oneline
  This is a shorthand for "--pretty=oneline --abbrev-commit" used together.
```

*Figura 5: significado del comando **–oneline**.*

```
• oneline

  <hash> <title line>

  This is designed to be as compact as possible.
```

*Figura 6: por qué fue “diseñado” el comando **–oneline**.*

4. Tome un screenshot del archivo gatos.py resultante corriendo en su computadora (1 punto).

```
stevonsa@stevonsa-VirtualBox:~/Git/LaboratorioGit/gatosGit$ python3 gatos.py
Caramelo, 4 annos, color: negro
Pelusa, 3 annos, color: cafe
Limón, 2 annos, color: blanco
stevonsa@stevonsa-VirtualBox:~/Git/LaboratorioGit/gatosGit$ cat gatos.py
import funcionesTarea

gato1 = funcionesTarea.gato()
gato1.nombre = "Caramelo"
gato1.color = "negro"
gato1.edad = 4
gato1.aniadirlista()

gato2 = funcionesTarea.gato()
gato2.nombre = "Pelusa"
gato2.color = "cafe"
gato2.edad = 3
gato2.aniadirlista()

gato3 = funcionesTarea.gato()
gato3.nombre = "Limón"
gato3.color = "blanco"
gato3.edad = 2
gato3.aniadirlista()

funcionesTarea.printlista()
```

Figura 7: screenshot del archivo.py siendo ejecutado.