TTPS Opción Ruby

Práctica 1: git

Primeros pasos

Antes de comenzar esta práctica es necesario tener git instalado. Asegurate que tengas git instalado ejecutando el siguiente comando:

```
$ sudo apt-get install git
```

Obteniendo ayuda

Para cualquier comando de git podés usar el subcomando help. Por ejemplo:

```
$ git help status
```

nos va a mostrar las opciones del subcomando status.

Otro punto a tener en cuenta es la salida del subcomando **status**. Dicha salida es sumamente detallada respecto de las acciones que se pueden hacer. Prestale atención!

Estados de git

¿Cuales son los 3 estados? ¿Qué significa cada uno?

Setup inicial

Antes de hacer nada, es necesario hacer un setup inicial de git. En dicho setup, vamos a decirle a git nuestra identidad y cuales son nuestras herramientas de preferencia. En primer lugar, vamos a decirle a git cual es nuestro nombre:

```
$ git config --global user.name “Nombre Completo”
```

y nuestro email:

```
$ git config --global user.email “nombrecompleto@gmail.com”
```

Luego, le vamos a decir a git cual es nuestro editor de texto preferido:

```
$ git config --global core.editor vim
```

y nuestra herramienta de diff preferida:

```
$ git config --global merge.tool vimdiff
```

Como se puede observar, las preferencias fueron configuradas de forma global, lo que significa que pueden ser sobreescritas de forma local.

Creando un repositorio

1 - Crea un directorio llamado **first_repo**
2 - Inicializá un repositorio git en el directorio **first_repo**

Agregando archivos

3 - Crea el archivo **README.md** (la extensión **md** significa que el archivo está escrito en markdown) con el siguiente texto:

```
# First repo
```

4 - Agregá el archivo **README.md** al área de staging. **Ayuda:** mirá el subcomando **add**.

5 - ¿Cómo te das cuenta que el archivo está en el área de staging? **Ayuda:** mirá la salida del subcomando **status**.

Comitiendo los cambios

6 - Los archivos ya están agregados al área de staging, lo que queremos hacer ahora es crear un commit y guardarlo en el repositorio git. **Ayuda:** mirá el subcomando **commit**.

Viendo la historia

7 - ¿Cuál es el hash del commit recién creado? **Ayuda:** mirá el subcomando **log**.

Deshaciendo cambios

8 - Agregá un archivo más llamado **file.txt** y comití los cambios con el siguiente mensaje: “Agregando otro archivo”

9 - El commit anterior tiene un mensaje corto y poco descriptivo: cambialo. **Ayuda:** mirá la opción **--amend** del subcomando **commit**.

10 - Surgió otro problema con el commit anterior: ahora te olvidaste de agregar un archivo llamado **oops**. Agregá el archivo al commit anterior.

11 - Crea el archivo **file2.txt** y agregalo al área de staging. Luego de trabajar nos dimos cuenta que no queremos comitir ese archivo: sacalo del área de staging. **Ayuda:** mirá la salida del subcomando **status**.

12 - Modificá el archivo **README.md** con el siguiente contenido:

```
# Second repo
```

y agregá el archivo al área de staging.

13 - Resulta que las modificaciones realizadas en el punto anterior son incorrectas y no las queremos: revertí los cambios. **Ayuda:** mirá la salida del subcomando **status**.

Trabajando con remotos

14 - ¿El repositorio **first_repo** tiene algún remoto? ¿Cómo hacemos para ver los remotos? **Ayuda:** mirá el subcomando **remote**.

15 - Ahora queremos guardar el repositorio en github: subilo a github! **Ayuda:** Creá el repositorio en github y github te va a ayudar con los pasos a seguir.

16 - ¿Qué hace el subcomando **push**?

17 - ¿Qué hace el subcomando **pull**?

18 - ¿Qué hace el subcomando **fetch**?