Taller - Introducción de Git

Table of Contents

En este taller usted realizará una serie de actividades que le permitirán conocer los comandos más básicos del sistema de control de versiones git y su uso, en particular, para enviar las tareas de este curso.

1 Introducción

Git es un sistema descentralizado de control de versiones que permite consultar y mantener una evolución histórica de su código dentro de lo que se conoce como "repositorio", que es el lugar digital en donde usted mantendrá todo su código fuente. Git le permite usar branches que son muy útiles para realizar desarrollo paralelo (probar ideas, por ejemplo) sin modificar el código principal. Además facilita la colaboración remota, entre muchas otras cosas. En este curso usaremos git para manejar las versiones de los programas y todas las tareas, proyectos, etc serán entregados como un repositorio en git.

Para mayor información consulte https://git-scm.com/docs , https://www.atlassian.com/git/tutorials , http://rogerdudler.github.io/git-guide/ , etc.

Es posible sincronizar sus repositorios locales con repositorios remotos, por ejemplo en páginas como https://github.com o https://bitbucket.org y esto le da grandes ventajas. Por ejemplo, usted crea un repositorio en la Universidad, lo sincroniza con github, envía los cambios a github, y luego al llegar a casa usted descarga la versión actualizada a un repositorio en su casa, hace modificaciones, y las envía a github, y al volver a la Universidad actualiza su repositorio de la Universidad descargando las últimas versiones enviadas a github, y asi sucesivamente.

2 Creando un repositorio vacío

Cuando usted va a iniciar sus tareas de programación lo primero que debe hacer es crear un directorio y allí iniciar un repositorio. Se asume que iniciará con un repositorio vacío y local, sin sincronizar con otro repositorio. Primero, para crear el subdirectorio de trabajo, deberá ejecutar los siguientes comandos

```
mkdir repoexample
cd repoexample
```

Hasta este momento tendra un directorio complemtamente vacío que ni siquiera tiene archivos ocultos (lo puede ver con el comando `ls -la` . Para inicializar un repositorio git, ejecute el siguiente comando dentro de ese directorio

```
git init
```

Listo, ya tiene su primer repositorio. Se ha creado un directorio oculto llamado .git que contiene toda la información de su repositorio. Usted nunca debe tocar ese subdirectorio.

3 Configurando git

Git le permite configurar varias opciones de utilería (como los colores en la impresión a consola) asi como otras más importantes y obligatorias como el nombre y el email del autor de los diferentes commits o envíos al repositorio. La configuración puede ser local al repositorio (afecta al repositorio local y no a otros) o global (afecta a todos los repositorios del usuario). Para configurar el nombre y el email de forma global debe usar los siguientes comandos:

```
git config --global user.name <name>
git config --global user.email <email>
```

donde debe reemplazar `<name>` por su nombre entre comillas y `<email>` por su correo electrónico.

Otras configuraciones útiles pueden ser añadir colores a la impresión a consola

```
git config color.ui true
```

o configurar aliases

```
git config --global alias.st status
git config --global alias.co checkout
git config --global alias.br branch
git config --global alias.up rebase
git config --global alias.ci commit
```

4 Creando el primer archivo y colocandolo en el repositorio

Cree un archivo README.txt que contenga cualquier contenido descriptivo del repositorio (es una buena costumbre siempre tener un readme). Para añadirlo al repositorio usted debe tener en cuenta que en git existe un área temporal en donde se colocan los archivos que se van a enviar en el próximo envío, llamada la stagging area (comando ~git add … ~), y un comando que envía de forma definitiva estos archivos del stagging area al repositorio (~git commit … ~). Esto muy útil para separar commits en términos de modificaciones lógicas. Por ejemplo, si ha editado cuatro archivos pero un cambio realmente solamente a un archivo y el otro a los demás tres, tiene sentido hacer dos commits separados.

Para colocar un archivo en el stagging area y ver el status de su repositorio use el siguiente comando,

```
git add filename
git status
```

Para "enviar" de forma definitiva los archivos al repositorio (enviar se refiere a introducir las nuevas versiones en el sistema de información del repositorio) use el siguiente comando, y siempre acostúmbrese a usar mensajes descriptivos

```
git commit -m "Descriptive message of the changes introduced."
```

Si omite el mensaje y escribe solamente

```
git commit
```

se abrirá el editor por defecto para que escriba el mensaje allí.

5 Ignorando archivos : Archivo `.gitignore`

Se le recomienda crear un archivo en la raíz del repositorio, llamado `.gitignore` , que contenga los patrones de nombres de archivos que usted desea ignorar para que no sean introducidos ni tenidos en cuenta en el repositorio. Un contenido típico es

```
*~
*.x
*a.out
```

6 Configurando el proxy en la Universidad

La universidad tiene un sistema proxy que filtra las salidas de internet cuando usted conectado a la red de cable de la Universidad o a la red wireless autenticada (no aplica si está conectado a la red de invitados). La configuración que se le indicará en este apartado le permitirá sincronizar los repositorios remotos y locales desde la consola usando la red cableada o wireless autenticada. Tenga en cuenta que esto no tiene nada que ver con la configuración del proxy del navegador que aunque parezcan similares no tienen nada que ver.

Debe exportar el proxy https para poder usar la consola y la sincronización, de manera que debe usar estos comandos (se le recomienda que lo anote en algo que mantenga con usted)

```
export https_proxy="https://username:password@proxyapp.unal.edu.co:8080/"
```

en donde `username` es el nombre de usuario de la universidad y el password es el password de la universidad. Siempre debe ejecutar este comando antes de poder clonar, sincronizar, enviar, traer, etc cambios desde un repositorio remoto.

7 Sincronizando con un repositorio remoto

7.1 Configurando la dirección del repositorio remoto (se hace una sola vez)

Usted tiene varias opciones para sincronizar con un repositorio remoto.

La primera opción es suponer que usted ha creado un repositorio local (como se ha hecho en este ejercicio), ha creado uno remoto (en la página de github) y desea sincronizar el repositorio local con el remoto. En este caso deberá indicarle a su repositorio local cuál es la dirección del remoto, usando el comando

```
git remote add origin <serverrepo>
```

Cuando usted crea un repositorio en github debe leer las instrucciones que salen ahí y que le explican claramente lo que debe hacer.

La segunda es clonar el repositorio remoto en su computador local (no dentro de un repositorio local sino dentro de una carpeta normal) usando los siguientes comandos:

```
git clone /path/to/repository
git clone username@host:/path/to/repository
```

Al usar el segundo comando usted habrá clonado el repositorio remoto en el disco local, y ese repositorio local tendrá información que le permitirá enviar los cambios al repositorio remoto.

7.2 Enviando cambios al repositorio remoto

Una vez tenga configurado el repositorio remoto puede enviar los cambios locales usando el comando

```
git push origin master
```

Este comando sincronizará el repositorio local con el remoto (sincronizará el branch master).

7.3 Trayendo cambios desde el repositorio remoto

Para traer cambios desde el repositorio remoto puede usar el siguiente comando

```
git pull
```

Este comando realmente ejecuta dos: `git fetch`, que trae los cambios, y `git update` que actualiza el repositorio con esos cambios. Si no existe ningún conflicto con la versión local los dos repositorios se sincronizarán. Si existe conflicto usted deberá resolverlo manualmente.

8 Consultando el historial

Una de las ventajas de git es tener un track de todos los cambios históricos del repositorio, que usted puede consultar con el comando

```
git log
```

Si desea ver una representación gráfica, use el comando `gitk` .

9 Otros tópicos útiles

En git usted puede hacer muchas cosas más. Una de ellas es usar branches. Las branches son como trayectorias de desarrollo paralelas en donde usted puede modificar código sin afectar a otras branches. El branch por defecto se llama master. Si usted quiere probar una idea pero no esté seguro de que funcione puede crear una nueva branch experimental para hacer los cambios allí, y en caso de que todo funcione puede hacer un merge de esa branch a la branch principal. Para crear branches puede usar el comando

```
git branch branchname
git checkout branchname
```

Para volver al branch master usa el comando

```
git checkout master
```

10 Práctica online

En este punto usted conoce los comandos básicos de git. Se le invita a que realice el siguiente tutorial que le permite practicar en la web Lo aprendido y otras cosas: https://try.github.io/levels/1/challenges/1 . **No pase a los siguientes temas sin haber realizado el tutorial interactivo**.

11 Enviando una tarea

Para enviar una tarea/proyecto/examen el profesor debe enviarle un link de invitación, por ejemplo, a su correo, y usted debe

1. Abrir un navegador y entrar a SU cuenta de github.
2. Después de haber ingresado en SU cuenta de github, abrir el enlace de invitación que le envió el profesor.
3. Debe aparecer una pantalla en donde hay un botón que dice "Accept this assignment". Dele click a ese botón.
4. Después de un pequeño tiempo aparecerá un mensaje de éxito y se le indicará una dirección web en donde se ha creado su repositorio, algo como https://github.com/SOMENAME/reponame-username . De click en ese enlace.
5. Ese es su repositorio para enviar la tarea y será el repositorio que se va a descargar. Lo normal es que usted haya arrancado localmente y quiera sincronizar el repositorio local con ese repositorio remoto de la tarea. Para eso siga las instrucciones que se explicaron más arriba sobre cómo añadir un repositorio remoto (básicamente debe las instrucciones que le da github en la parte que dice **"… or push an existing repository from the command line"**) .

6. No olvide que cada vez que haga un commit el repositorio local también debe enviar esos cambios al repositorio remoto, usando el comando

   ```
   git push -u origin master
   ```

12 Ejercicio

El plazo máximo de entrega es **<2017-02-03 Fri 15:00>** . Cualquier envío posterior será ignorado.

1. Cree un repositorio local (inicialmente vacío).
2. Cree y añada al repositorio un archivo `README.txt` que contenga su nombre completo y código, junto con una descripción sencilla de la actividad. (en este punto debe haber por lo menos un commit).
3. Cree un archivo `.gitignore` para ignorar archivos que no sean importantes y añádalo al repositorio. (otro commit))
4. Acepte la invitación de la tarea y sincronice el repositorio de la tarea con el local. Envíe los cambios que ha hecho hasta ahora.
5. Escriba un programa en `C++` que lea un entero y que imprima el número multiplicado por dos. El programa se debe llamar `mult.cpp`. El operador que se usa para multiplicar es el *. Añada este programa al repositorio.
6. Escriba un programa en `C++` que pregunte su nombre e imprima un saludo a su nombre como "Hola, William!". El programa se debe llamar `hola.cpp`. Para eso revise la presentación de introducción que se inició la clase pasada. Añada este programa al repositorio.
7. Investigue sobre los tipos de datos `int, float, double, char` en `C++` y escriba una descripción breve de cada uno dentro de un archivo llamado `datatypes.txt` que también deberá añadir al repositorio. Por ejemplo, investigue por el uso de cada tipo de datos, cómo se declaran, cuáles son los límites mínimos y máximos, etc. Añada este programa al repositorio.
8. Con lo anterior escriba un un programa en `C++` que lea de la pantalla un número de punto flotante de precisión doble y que imprima el número divido por 3. El operador que se usa para dividir es / . Añada este programa al repositorio.
9. No olvide sincronizar el repositorio local con el remoto, antes de la hora límite.

Author: William Oquendo, woquendo@gmail.com

Created: 2017-02-02 Thu 21:41

Validate