MT SOFTECH

SOLUCIONES EN INFORMÁTICA

Comandos de Git

Autor : Ricardo Delgado

Comandos de Git

Git es un sistema de control de versiones gratuito y de código abierto que nos permite gestionar nuestros proyectos de una manera rápida, ordenada y eficiente, mediante el uso de comandos específicos que tienen como propósito llevar un correcto registro de cambios a lo largo del proyecto, coordinando de esta manera el trabajo realizado por varios integrantes del equipo.

Esta herramienta es fácil de aprender, y solo basta con memorizar los comandos o acciones que se describirán a continuación para poder comenzar a colaborador con los distintos proyectos o repositorios de nuestro equipo.

Cabe destacar que git se maneja por medio de repositorios (lo que sería equivalente a una carpeta en la computadora), así que está palabra aparecerá constantemente en las definiciones proporcionadas.

Comandos

Configuración básica:

Cuando es la primera vez que utilizamos git, es necesario configurar las credenciales que usaremos en nuestro equipo, así que debemos ejecutar los siguientes comandos:

Configurar el nombre de usuario:

```
git config --global user.name "mi_usuario"
```

Configurar el correo electrónico

```
git config --global user.email mi_correo@correo.com
```

Iniciar repositorio:

Se realiza la primera vez que vamos a interactuar con el repositorio.

```
git init
```

Clonar repositorio:

Está acción se realiza cuando una persona nos invita a colaborar en su proyecto y necesitamos descargar/obtener los archivos que se encuentran en el repositorio. Como consecuencia, obtendremos una carpeta con los archivos del repositorio.

```
git clone https://pagina_git/usuario/nombre_proyecto.git
```

Otra manera de clonar nuestro repositorio y, además, que este nos pida nuestras credenciales, es la siguiente:

```
git clone https://usuario@pagina_git/usuario/nombre_proyecto.git
```

Decidir que archivos tomar en cuenta para realizar un cambio:

Cuando le hemos hecho alguna modificación al proyecto, es necesario elegir los archivos que queremos seleccionar para que los demás sepan que esos archivos serán modificados. Comúnmente esta acción se realiza cuando hemos acabado de realizarle los cambios al proyecto por el día de hoy.

Seleccionar todos los archivos modificados:

```
git add .
```

Seleccionar un archivo en específico:

```
git add midocumento.txt
```

Guardar los cambios realizados en nuestro equipo:

Este comando es uno de los más utilizados, ya que es el que establece un punto de control en el proceso de desarrollo, al cual podemos volver más tarde si así lo deseamos. Suele identificarse con un mensaje corto explicando los cambios que se han realizado en el repositorio.

```
git commit -m “mi mensaje”
```

Mostrar los nombres y detalles de los commits:

Una forma rápida de mostrar todos los commits realizados en un proyecto con sus respectivos detalles es la siguiente:

```
git log
```

Revertir un commit:

En ocasiones sucede que es necesario revertir ciertos cambios por varias situaciones, ya sea alguna equivocación o algún cambio que no planeábamos subir, más sin embargo ya hicimos un commit. Una manera segura de revertirlo es primero viendo los detalles de nuestro commit (comando anterior) y después ejecutando el siguiente comando:

```
git revert numero_de_commit
```

Subir los cambios a nuestro gestor de versiones:

Una vez que hemos confirmado los cambios a realizar, el siguiente paso es enviar estos cambios al servidor remoto para que los demás integrantes puedan verlos, obtenerlos y seguir trabajando.

Si es el primer cambio del proyecto, será necesario ejecutar:

```
git push -u origin nombre_de_tu_rama
```

En cambio, si ya se han hecho varios commits, bastaría con ejecutar:

```
git push nombre_remoto nombre_de_tu_rama
```

O en una versión más simple:

```
git push
```

Bajar los cambios de los demás integrantes:

Ahora bien, así como puedes subir tus cambios también es indispensable obtener las modificaciones que han hecho los demás, con el fin de estar actualizado siempre en cuanto al proyecto.

```
git pull
```

Agregar nuestro propio repositorio:

Por otro lado, cuando nosotros somos los creadores del repositorio será indispensable configurarlo con nuestra url en lugar de clonar el proyecto.

Agregar un nuevo repositorio:

```
git remote add origin mi_url
```

Cambiar el repositorio:

```
git remote set-url origin mi_url
```

Eliminar el repositorio:

```
git remote rm nombre_repositorio
```

Mostrar lista de repositorios:

```
git remote -v
```

Ramas (branches):

Una rama es un espacio de trabajo independiente en el cual uno o varios miembros pueden trabajar. La diferencia de utilizar una o varias ramas es que se puede organizar de mejor manera el proyecto, ya sea por módulos, áreas de trabajo, etc. Sin embargo, cuando se utiliza más de una rama será necesario realizar una unión (siguiente comando) para integrar el código de todas las ramas en uno solo.

Agregar una rama:

```
git branch nombre_de_la_rama
```

Cambiar de rama:

```
git checkout nombre_de_la_rama
```

Obtener información de rama actual:

```
git status
```

Mostrar todas las ramas:

```
git branch
```

Eliminar el repositorio:

```
git remote rm nombre_repositorio
```

Eliminar una rama y unir el contenido a la rama maestra (master):

```
git branch -d nombre_de_la_rama
```

Eliminar una rama definitivamente (sin uniones ni confirmaciones):

```
git branch -D nombre_de_la_rama
```

Unir todo el código en uno solo:

Cuando se maneja más de una rama o un integrante del equipo no ha bajado los cambios más recientes y planea subir modificaciones nuevas que afectan a estos últimos cambios, es necesario ejecutar este comando para unir todo el código en uno solo.

```
git merge nombre_de_la_rama
```