**GIT y MERCURIAL**

Para empezar:...

Tanto MERCURIAL como Git son sistemas de manejo o control de versiones para el código de cualquier proyecto de Software. Por medio de estos programas se tiene un completo seguimiento o historial de las modificaciones realizadas en cada archivo y su código. De manera que nos permiten revisar los cambios realizados, regresar a versiones anteriores, generar espejos de nuestros proyectos(muy útil cuando se trabaja en un proyecto con equipo de desarrollo compuesto por varios programadores) y así tener un respaldo mucho mas efectivo y ordenado, de todas las versiones de nuestro código gracias a los “commit”.

**Instalación de GIT:...**

Sobre la terminal verifico si ya está instalado con:..

* **git --version**

Si no está instalado, ejecuto el siguiente comando:..

* **Sudo apt-get install git**
* Después del proceso, ejecuto:.. **git --version** y me debe mostrar la versión que tengo instalada, que para este caso es 2.17.0

**Generalidades de GIT:...**

En Git siempre se traban los proyectos en base a TRES estados, que son:..

1. Working-Directory:

Es donde se editan o modifican los proyectos

1. Stagin-Area:

Estado en el que seleccionamos los archivos que ya están listos para el tercer estado, pero también escogemos los archivos que aún no están listos para dicho estado

1. Repository:

En este nivel o estado se registra todo lo realizado en los archivos que pasan a este nivel, por medio de un proceso llamado commit.

Así. El proceso de trabajo en GIT, lo podemos generalizar como:...

**3**

**1**

**2**

**Crear o editar Archivos Filtrar Archivos Registrar Proyecto o cambios**

Vamos a realizar unas configuraciones iniciales en GIT:...

**Configurar el nombre de usuario**, con el comando:..

**git config --global user.name "nombre\_de\_usuario"**

**Para ver esa configuración**, ejecutamos:...

**git config --global user.name**

Configurar e-mal:...

**git config --global user.email beanswalker.io@gmail.com**

**Configurar colores** para los resultados(es como una semaforización):..

**git config --global color.ui true**

**Ver el estado de las anteriores configuraciones:..**

**git config --global --list**

**Crear proyecto en GIT:...**

En mi directorio de usuario(carpeta personal), he creado el directorio Development/git\_projects/proyecto1 con el explorador de archivos. Ahora, mediante la terminal me ubico en dicho directorio y ejecuto el comando:...

**git init**

El anterior comando inicializa un repositorio en ese directorio, de manera que GIT empieza a monitorear todos los cambios realizados a partir de ese momento

**Status:..**

El siguiente comando nos informa el estado de nuestro proyecto y sobre que copia o versión del proyecto nos encontramos:..

**git status**

Si me dirijo mediante el explorador de archivos y creo directorios y/o archivos en el directorio master(proyecto1). Al volver a ejecutar **git status,** el sistema advierte que en el proyecto se han agregado archivos, pero que no se ha dado la orden para hacer seguimiento a dichos archivos.

**Iniciar seguimiento a archivo**:..

Para ejecutar el inicio de seguimiento a un archivo, ejecutamos:...

**git add nombre\_archivo.extensión**, por ejemplo...

**git add index.php**

**Iniciar seguimiento a todo el contenido del directorio master**:..

Para ejecutar el inicio de seguimiento a TODOS los archivos del proyecto, ejecutamos:...

**git add -A**

**Agregar los cambios realizados a algún o algunos archivos**:...

**git commit -m “ingresamos un mensaje a modo de comentario”**

**Ver historial de los commit realizados:...**

**git log**

**Viajar en el tiempo, hacia una versión anterior del archivo:...**

**Después de haber ejecutado git log, copiamos la llave generada por git, del commit al cual queremos regresar, y ejecutamos:...**

**git checkout llave\_generada por ejemplo:**

**Git checkout d12096ebe72e73603cfe9a9b2346557aca369d33**

**Asi. El proyecto regresa al estado especificado.**

**Regresar a la versión mas actualizada del proyecto:...**

**git checkout master**

**Poner como master una versión anterior de manera definitiva:..**

* git reset --hard código\_hash\_de\_la\_versión\_deseada. El problema Es que hay que tener MUCHO cuidado porque de esta manera se borran TODOS los commit realizados después de la versión seleccionada para master.

**Git y Github desde GitKraken:...**

**Titulo añadido por developer y no por master**

**Instalación de MERCURIAL:...**

Sobre la terminal verifico si ya está instalado con:..

* hg --version

Si no está instalado, ejecuto el siguiente comando (o lo instalo mediante synaptic):...

* Sudo apt-get install mercurial
* Después del proceso, ejecuto:... hg --version y me debe mostrar la versión que tengo instalada, que para este caso es 4.6