Tutorial GIT, GitHub

Introducción a GIT, GitHub
Autor: Luis Daniel Benavides Navarro

GIT

¿Qué es GIT?

- Un sistema de control de versiones
  - Un sistema que almacena cambios sobre un archivo o un conjunto de archivos
  - Un sistema que permite recuperar versiones previas de esos archivos
  - Permite otras cosas como el manejo de ramas (Branches)

Tipos de sistemas de control de versiones

- Sistema Local de manejo de versiones (e.g., RCS aún distribuido en los MACs)
- Sistema Centralizado de manejo de versiones (Subversion)
- Sistema Distribuido de manejo de versiones (GIT)

Sistema Local de manejo de versiones (e.g., RCS aún distribuido en los MACs)

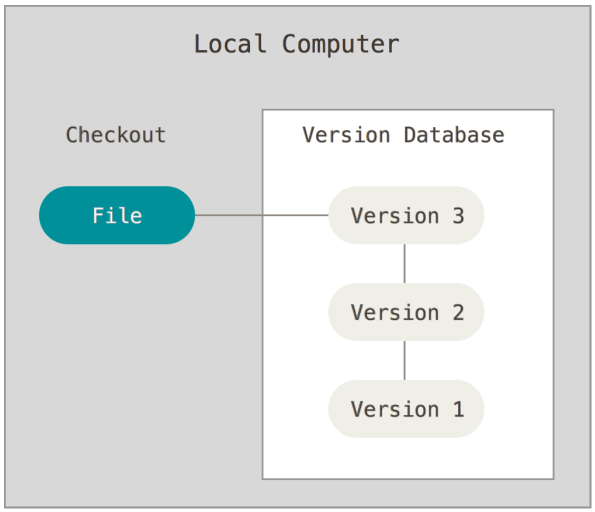


Imagen extraída del sitio web de Git (www.git-scm.com)

Sistema Centralizado de manejo de versiones (Subversion)

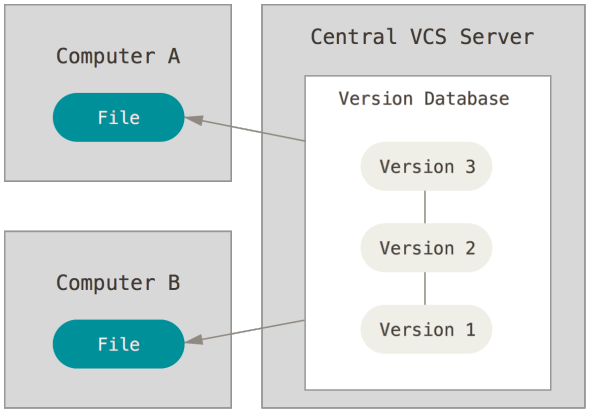


Imagen extraída del sitio web de Git (www.git-scm.com)

Sistema Distribuido de manejo de versiones (GIT)

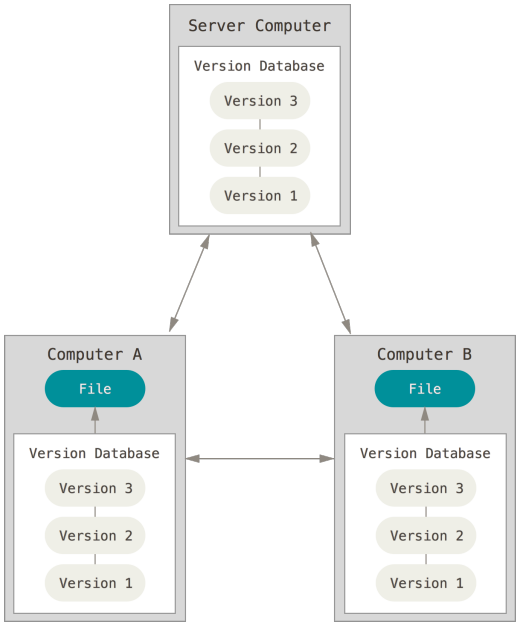
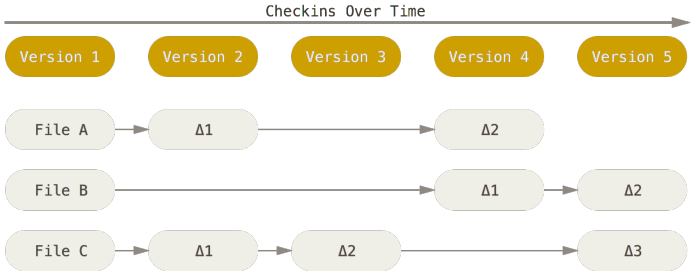


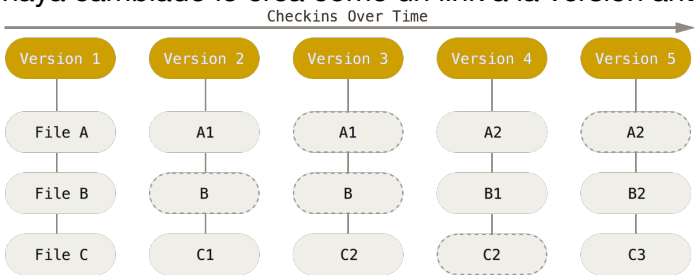
Imagen extraída del sitio web de Git (www.git-scm.com)

¿Cómo almacena datos GIT?

Los VCS tradicionales, manejan los datos como el archivo original y una serie de diferencias que son almacenadas en la base de datos



GIT almacena cada versión como una fotografía de todos los archivos, y el archivo que no haya cambiado lo crea como un link a la versión anterior.

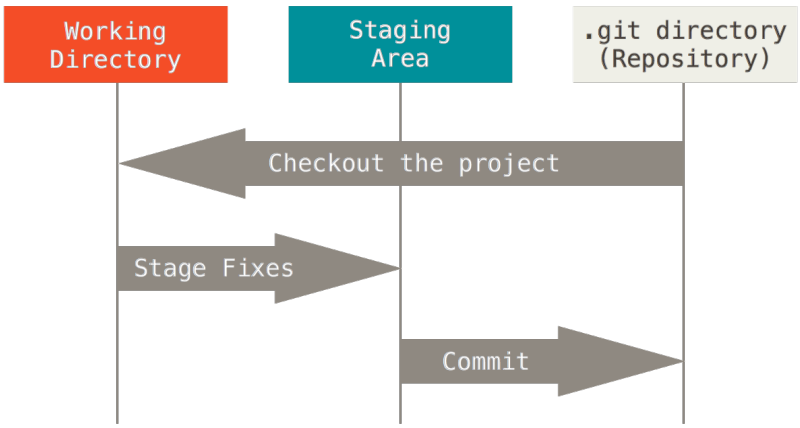


Otras características

- La mayoría de operaciones son locales
- Git tien integridad: todo es revisado por un hash antes de ser almacenado
- Git solo adiciona datos,i.e., es muy difícil hacer algo que no se pueda deshacer

Los tres estados de un archivo en GIT (IMPORTANTISIMO)

- Commited: su archivo esta seguro almacenado en su repositorio LOCAL
- Modified: Archivo modificado pero aun no Commited
- Staged: Archivo marcado para ser cometido en la siguiente versión del sistema



El flujo de trabajo típico de GIT

1. Modificar archivos en su directorio de trabajo
2. Marcar los archivos como staged
3. Hacer un commit para pasar los archivos de Commited a Staged

Instale GIT

- Siga las instrucciones en el sitio web de GIT
  - http://git-scm.com/book/en/v2/Getting-Started-Installing-Git
- ```
  mbp-de-luis-2:~ dnielben$ git --version
  git version 2.2.1
  ```

Configure su identidad y su editor por defecto

- $ git config --global user.name "John Doe"
- $ git config --global user.email johndoe@example.com
- git config --global core.editor emacs
- Revise su configuración
  - ```
    git config --list
    ...
    user.name=Daniel Benavides
    user.email=dnielben@gmail.com
    core.editor=emacs
    ```

¿Cómo obtener ayuda?

- git help <verb>
  $ git <verb> --help
  $ man git-<verb>
- Ejemplo
  - $ git help config
- El libro en línea en el sitio web de GIT

Crear un repositorio GIT

- En el directorio de su proyecto escriba
  - `$ git init`
- Obtendrá el siguiente mensaje
  - `Initialized empty Git repository in /Users/dnielben/NetBeansProjects/mi-primera-app/.git/`
- *Ha creado un repositorio local de Git!*

Agregar archivos al control de versiones de su repositorio

- Use el comando "add"
  - `$ git add pom.xml`
    `$ git status`
- Obtendrá la siguiente salida

```
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   pom.xml

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        src/
        target/
```

Confirmar los cambios en GIT y crear una nueva versión (commit)

- Para crear una nueva versión y confirmar los cambios (commit) escriba
  - `$ git commit -m 'Primera versión del proyecto'`
- Obtendrá la siguiente salida
  - `[master (root-commit) b75ab28] Primera versión del proyecto`
  - `1 file changed, 40 insertions(+)`
  - `create mode 100644 pom.xml`
- Si utiliza el comando de “git status” obtendrá algo similar a

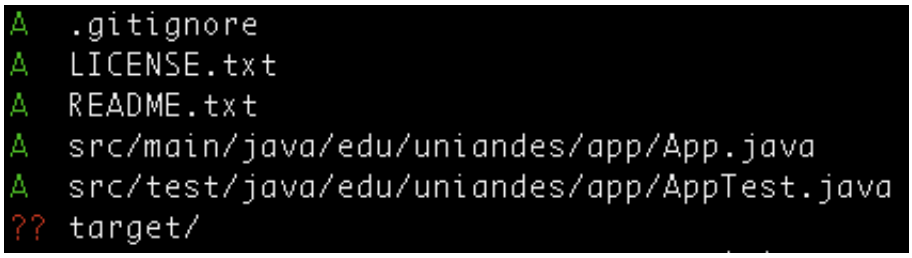
```
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        src/
        target/

nothing added to commit but untracked files present (use "git add" to track)
```

Trabajando con repositorios remotos (remotes)

- Para saber el listado de directorios remotos con los que ha estado trabajando digite
  - `$git remote`
- Por ahora no hay ningún repositorio configurado
- **Diríjase al sitio web de Github y cree un repositorio nuevo en GIthub**
- En su su proyecto git local adicione el repositorio remoto que acaba de crear con el nombre “origin"
  - `$ git remote add origin https://github.com/dnielben/miprimerrep.git`
- Revise que el repositorio de nombre origin esta en su listado con “git remote"
- Revise que las urls de los repositorios remotos están correctas con “git remote -v"
- Ahora empuje la ultima versión del proyecto que tiene almacenada en su computador
  - `git push -u origin master`
- Obtendrá una salida similar a:
  - `Username for 'https://github.com': suusuario`
  - `Password for 'https://suusuario@github.com':`
  - `Counting objects: 3, done.`
  - `Delta compression using up to 4 threads.`
  - `Compressing objects: 100% (2/2), done.`
  - `Writing objects: 100% (3/3), 624 bytes | 0 bytes/s, done.`
  - `Total 3 (delta 0), reused 0 (delta 0)`

    - `To https://github.com/dnielben/miprimerrep.git`
    - `* [new branch] master -> master`
    - `Branch master set up to track remote branch master from origin.`
- Ahora ya esta sincronizado su repositorio local con su repositorio remoto!
- Su rama (brach) de trabajo por defecto es “master” y su repositorio remoto por defecto es “origin"

Agregar más archivos a su proyecto de Git y su repositorio remoto

- Es una práctica recomendada crear un archivo README, otro LICENSE y otro .gitignore
    - `$ echo ‘Mi primer proyecto’ > README.txt`
    - `$ echo ‘TODO: Copiar el texto de la licencia http://www.gnu.org/licenses/gpl.html' > LICENSE.txt`
    - `$ echo ‘# TODO: Copiar los ontenidos de https://github.com/github/gitignore/blob/master/Java.gitignore' > .gitignore`
- Adicione los archivos
    - `$ git add *.txt`
    - `$ git add .gitignore`
- OJO: Git ignore puede incluir expresiones glob, para indicarle a git que archivos ignorar
- Adicione el directorio src y recursivamente todos los contenidos de este
    - `$ git add src`
- Revise el estado de su proyecto (la opción “-s” muestra una lista compacta)
    - $ git status -s
- Obtendrá algo así:
    - 

Ignorando algunos archivos

- Modifique su archivo .gitignore para que no considere el directorio target
    - agregue las siguientes líneas

        - `# Ignora todos los archivos en el directorio target en los proyectos creados con maven`
        - `target/`
- Revise nuevamente el status de su proyecto
    - `$ git status -s`
- Obtendrá la siguiente salida
    - 

- Mire que el archivo “.gitignore” que está en el área staged es diferente al modificado
- Entonces debemos agregarlo nuevamente al area staged
    - `git add .gitignore`
- y obtendrá
    - 

Realice ahora el commit y el push remoto

- Ahora realice el commit
    - `git commit -m 'Segunda versión del proyecto adicionando e ignorando más archivos'`
- Ahora empuje los nuevos cambios al repositorio remoto
    - $ git push
- Si alguien modificara el repositorio, usted puede obtener los últimos cambios con
    - $ git pull
- Este comando trae los cambios y los combina con los cambios que usted tenga

Obtener una copia de un repositorio remoto

- OJO, intente esto en otro directorio diferente a su proyecto
- Puede obtener una copia de un repositorio usando el comando "clone"
- Intente bajar su proyecto en otro directorio
    - `$ git clone`

```
https://github.com/dnielben/miprimerrep copiademirep
```

- Ahora tiene una copia de su repositorio totalmente independiente, el trabajo en ada una será independiente y tendrá que sincronizarlos por medio del servidor central en github.
- En este proyecto usted ya podría ejecutar los comandos de maven
  - `$ mvn package`

Otros comandos útiles

- Ver exactamente que cambios(lineas) hizo y aun no ha enviado al area *staged*
  - `$ git diff`
- Ver exactamente que cambios(lineas) hizo y aun no ha enviado al repositorio local (commit)
  - `$ git diff --staged`
- Cuando quiero hacer commit de todo lo que he modificado y ya esta siendo seguido por git pero sin parar por el area staged
  - `$ git commit -a -m ‘commit directo sin pasar por staged'`
- Remover archivos del proyecto y del control de Git
  - Remover un archivo del directorio pero no del control de Git
    - $ rm [archivo]
  - Remover archivo del control de Git, primero “rm” lo pasa a staged y el commit lo confirma
    - $ git rm [archivo]
    - $ git commit -m ‘Confirmar la eliminación del archivo'
  - Remover un archivo del area staged
    - $ git rm —cached [archivo]
  - Puede usar patrones glob para remover cosas
    - `$ git rm log/ \*.log` (Remueve los archivos que terminan con log en el directorio log)
    - `$ git rm \*~` (Remueve todos los archivos que terminan con ~)
  - Renombrar o mover archivos
    - `$ git mv file_from file_to`
  - Git también se da cuenta si mueve un archivo de la siguiente manera, se da cuenta que es el mismo archivo
    - `$ mv README.md README`
    - `$ git rm README.md`
    - `$ git add README`
  - Ver el historial de commits
    - `$ git log`