Ingeniería del Software

1. Introducción a Git, Markdown y Eclipse

Antonio Manuel Durán Rosal

Asignatura "Ingeniería del Software"
2º Curso Grado en Ingeniería Informática
Escuela Politécnica Superior
(Universidad de Córdoba)
aduran@uco.es

18 de septiembre de 2018

Contenidos I

1.1. Git.
- 1.1.1. Introducción.
- 1.1.2. Instalación y configuración.
- 1.1.3. Uso básico.
- 1.1.4. Ramas.
- 1.1.5. Github.

1.2. Markdown.
- 1.2.1. Introducción.
- 1.2.2. Código.

1.3. Eclipse.
- 1.3.1. Introducción.
- 1.3.2. Instalación.

1.4. Recursos.

Evaluación

- Las entregas en moodle se realizarán por medio del representante o líder de cada grupo.
- Se debe entregar en moodle la dirección del repositorio de Github.
- Se evaluará la realización de un pequeño tutorial de Github con los contenidos aprendidos durante las dos primeras sesiones prácticas. El lenguaje de formateado será Markdown.
- El repositorio de Github contendrá tanto el tutorial como el historial de cambios realizados por los integrantes del grupo.

Motivación

- Código efímero.
- Necesidad de mantener todas las versiones del código fuente.
- Problemas en organizaciones para mantener el código actualizado.
- Coherencia de versiones.
- Conocimiento del cambio que ha provocado que el sistema no funcione.
- Fallos en el disco duro que suponen riesgo de información desactualizada.
- Satisfacer el compromiso de entrega.

Git y GitHub

Git

: sistema para el control distribuido de versiones de código. Fundamentalmente permite:

- Dar seguimiento a los cambios realizados sobre un archivo.
- Almacenar una copia de los cambios.

GitHub

: sitio web dónde podemos subir una copia de nuestro repositorio Git.

Ventajas

Git

- Habilidad de deshacer cambios.
- Historial y documentación de cambios.
- Múltiples versiones de código.
- Habilidad de resolver conflictos entre versiones de distintos programadores.
- Copias independientes.

GitHub

- Documentación de requerimientos.
- Ver el avance del desarrollo.

Instalación

- Para instalar Git: https://git-scm.com.
- En el curso se utilizará Git a través de líneas de comandos.
- Para eclipse existen *plugins* integrados: https://www.eclipse.org/egit.

Configuración básica

Nombre del administrador:

```
git config --global user.name "Antonio M. Durán Rosal"
```

Correo electrónico:

```
git config --global user.email aduran@uco.es
```

Editor de texto:

```
git config --global core.editor "gedit"
```

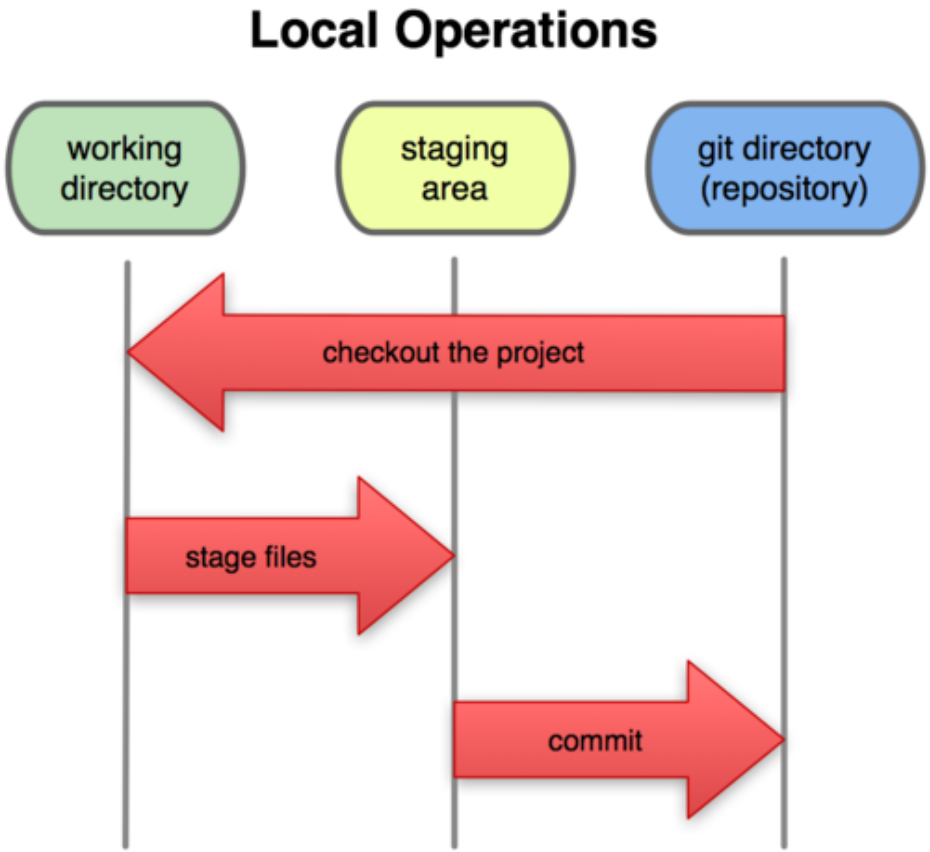
Color de la interfaz:

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

Listado de la configuración:

```
git config --list
```

Los tres estados de Git



Comandos básicos I

Iniciar repositorio en un directorio:

```
git init
```

Agregar cambios al area de *staging*:

```
git add
```

Validar cambios en el repositorio:

```
git commit -m "Mensaje"
```

Hacer los dos pasos anteriores en uno:

```
git commit -am "Mensaje"
```

Historial de commits:

```
git log
```

Comandos básicos II

Ayuda del listado anterior:

```
git help log
```

Listar los 5 commits más recientes:

```
git log -n 5
```

Listar los commits desde una fecha:

```
git log --since=2018-09-18
```

Listar los commits por autor:

```
git log --author="Antonio"
```

Ver cambios en el directorio:

```
git status
```

Comandos básicos III

Ver diferencia entre ficheros en el directorio y el repositorio de git:

```
git diff
```

Ver diferencia entre ficheros en el *staging* y el repositorio:

```
git diff --staged
```

Eliminar archivos:

```
git rm archivo
git commit -m "Mensaje"
```

Mover o renombrar archivos:

```
git mv antiguo nuevo
git commit -m "Mensaje"
```

Comandos básicos IV

Deshacer cambios con git:

```
git checkout -- nombre_fichero
```

Retirar archivos del *staging*:

```
git reset HEAD nombre_fichero
```

Complementar último commit:

```
git commit --amend -m "Mensaje"
```

Recuperar version de un fichero de commit antiguo:

```
git checkout <id_commit> -- nombre_archivo
```

Revertir un commit:

```
git revert <id_commit>
```

Comandos básicos V

Deshacer multiples cambios en el repositorio:

```
git reset --soft <id_commit>
git reset --mixed <id_commit>
git reset --hard <id_commit>
```

Listar archivos que git no controla:

```
git clean -n
```

Eliminar archivos que git no controla:

```
git clean -f
```

Ignorar archivos en el repositorio: .gitignore

Comandos básicos VI

Listar el contenido del repositorio de git:

```
git ls-tree master
git ls-tree master^^^
git ls-tree master~3
```

Log en una línea:

```
git log --oneline
```

Log con los tres últimos commits en una línea:

```
git log --oneline -3
```

Para más opciones consultar documentación de git.

Comandos básicos VII

Examinar el contenido de un commit:

```
git show <id>
```

Comparar un commit con el actual:

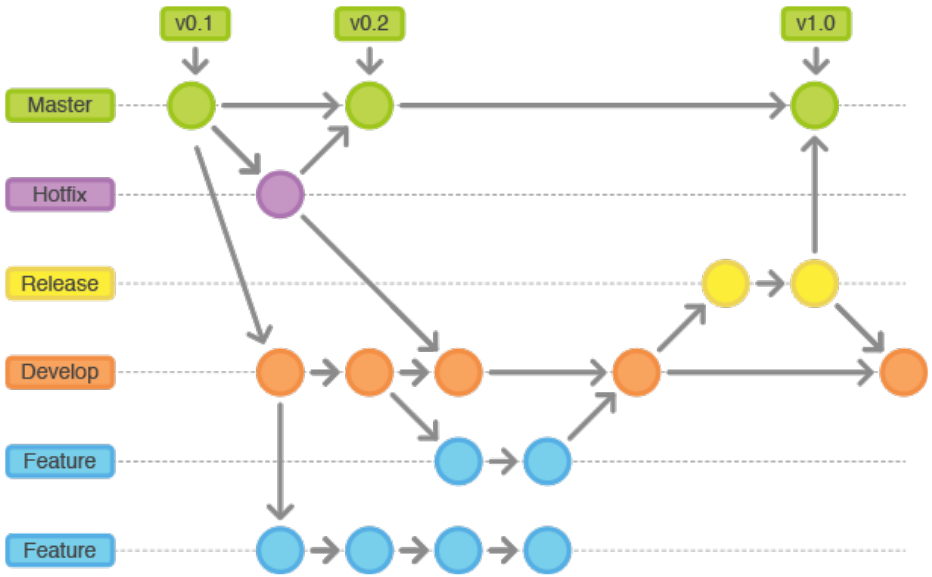
```
git diff <id> nombre_archivo
```

Comparar dos commits:

```
git diff id..id nombre_archivo
```

Ramas o *Branches*

Es la forma para separar la línea actual de desarrollo con respecto a la principal. Normalmente representan versiones del software que posteriormente son integradas a la línea principal.



Comandos Ramas I

Ver listado de ramas:

```
git branch
```

Crear una rama:

```
git branch nombre_rama
```

Cambiarnos a una rama:

```
git checkout nombre_rama
```

Crear una rama y moverse en un paso:

```
git checkout -b nombre_rama
```

Comparar ramas:

```
git diff nombre_rama..nombre_rama
```

Comandos Ramas II

Ver ramas idénticas a la actual:

```
git branch --merged
```

Renombrar ramas:

```
git branch -m nombre_antiguo nombre_nuevo
```

Eliminar ramas

```
git branch -d nombre_rama
git branch -D nombre_rama
```

Integrar ramas a la actual:

```
git merge nombre_rama
```

Resolver conflictos (se suele hacer manualmente):

```
git merge --abort
```

Comandos Ramas III

Almacenar cambios temporales:

```
git stash save "Mensaje"
```

Listar cambios:

```
git stash list
```

Ver contenido de un cambio temporal:

```
git stash show -p nombre_stash
```

Eliminar un cambio temporal:

```
git stash drop nombre_stash
```

Aplicar cambio del *stash*:

```
git stash apply nombre_stash
git stash pop nombre_stash
```

GitHub no es Git

Comandos GitHub I

Añadir repositorio remoto:

```
git remote add origin url
```

Ver repositorios remotos:

```
git remote -v
```

Eliminar repositorio remoto:

```
git remote rm origin
```

Añadir cambios del repositorio local al remoto:

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

Añadir cambios del repositorio remoto al local:

```
git pull
```

Comandos GitHub II

Ver *branches* remotos:

```
git branch -r
```

Ver todos los *branches*:

```
git branch -a
```

Clonar un repositorio remoto:

```
git clone url
```

Dar seguimiento a *branches* remotos

- LOCAL → REMOTO
  1. Cambios en el repositorio local.
  2. Commit de los cambios.
  3. Añadir cambios a repositorio remoto:

```
git push
```

- REMOTO → LOCAL
  - Sincronización y unión:

```
git fetch origin
git merge origin/master
```

  - En un solo paso:

```
git pull
```

Operaciones con *branches* remotos

- Creación:
  1. Crear branch local.
  2. Hacer cambios en dicho branch.
  3. Hacer commit.
  4. Copiar el branch al repositorio remoto:

```
git push -u origin branch_remoto
```

- Copia:

```
git checkout -b local remoto
```

- Eliminación:

```
git push origin --delete branch_remoto
```

Lenguaje Markdown

- Markdown es un lenguaje de etiquetado ligero que simplifica la elaboración de documentos.
- Se ideó pensando en una herramienta para escribir páginas web en un texto simple fácil de leer.
- Actualmente, se utiliza para documentar software ya que al ser texto plano puede entrar dentro de cualquier sistema de control de versiones e incluye muchas extensiones para colorear código fuente en distintos lenguajes.

Sintaxis I

Formato	Sintaxis
Negrita	**Texto en negrita**
Cursiva	*Texto en cursiva*
Lista con viñetas	1. Primera línea 2. Segunda línea
Lista anidada	* Primer nivel * Segundo nivel
Encabezados (hasta 6 niveles)	# Encabezado primer nivel # # Encabezado segundo nivel # # # Encabezado nivel tres
Citas en bloque	> Las citas en bloque deben comenzar y terminar con una línea en blanco.

Sintaxis II

Formato	Sintaxis
Código en línea	‘Esto es codigo en linea‘
Bloques de código	~~~ Ejemplo de bloque ~~~
Imágenes	![Texto alternativo](url/imagen.png)
Vínculos	[Texto del vínculo](url)
Imágenes con vínculos	[![Texto alternativo](imagen)](url)
Línea horizontal	- - - (Salto de línea antes y después)
Salto de línea	(Dos saltos de línea antes)

Eclipse

Eclipse es un entorno integrado de desarrollo (IDE).

- Se diseñó inicialmente como IDE para Java, sin embargo ahora soporta otros lenguajes como C++.
- Ayuda a escribir código más rápido y libre de algunos errores sintácticos, y ayuda a mantener un estilo de programación homogéneo.
- Facilita la depuración de código.
- Hay una gran documentación.

Instalación

- Utilizaremos eclipse para C++.
  Eclipse para C++
- Para eclipse existen *plugins* integrados con git.
  https://www.eclipse.org/egit

Recursos

Recursos Git:

- Guía sencilla de Git.
- Pro Git book.

Recursos Markdown:

- Markdown Cheatsheet.
- Guía en castellano extendida.

Recursos Eclipse:

- En las aulas se puede cargar con la orden eclipse3.3
- Eclipse para C++

Ingeniería del Software

1. Introducción a Git, Markdown y Eclipse

Antonio Manuel Durán Rosal

Asignatura "Ingeniería del Software"
2º Curso Grado en Ingeniería Informática
Escuela Politécnica Superior
(Universidad de Córdoba)
aduran@uco.es

18 de septiembre de 2018