

Nuevas tendencias en el diseño y desarrollo de aplicaciones con tecnologías Java



UNIVERSIDAD DE MÁLAGA



ANDALUCÍA TECH
Campus de Excelencia Internacional

accenture technology

Módulo 3: DevOps

Tema 10: Herramientas para el desarrollo de software

Loli Burgueño



Índice

- Gestión de código fuente con Git
- Construcción de código con Maven
- Integración continua en Jenkins



Gestión de código fuente

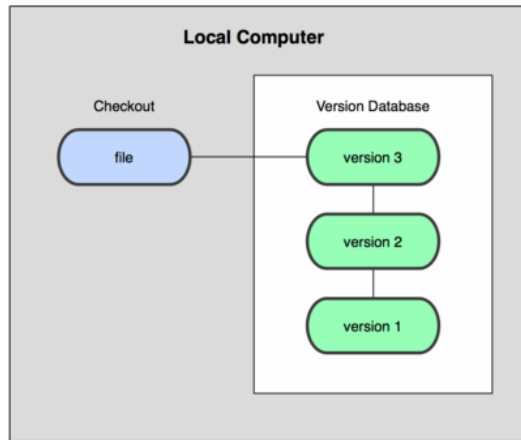
- ¿Qué es un sistema de control de versiones?
 - Herramienta que facilita la administración de las distintas versiones de un producto
- Propósito
 - Gestión de los cambios y modificaciones de:
 - Proyectos software (programas y ficheros de configuración)
 - Sitios Web
 - Documentación
- Fundamento
 - Se basan en la gestión de un repositorio



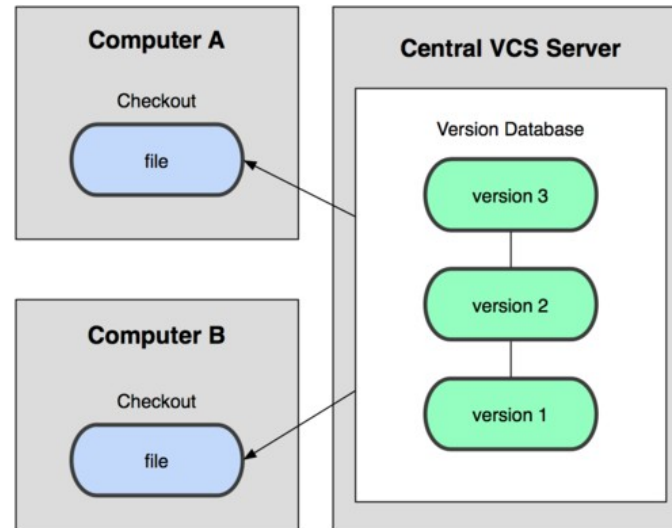
Tipos de sistemas de control de versiones

<http://git-scm.com/book/en/Getting-Started-About-Version-Control>

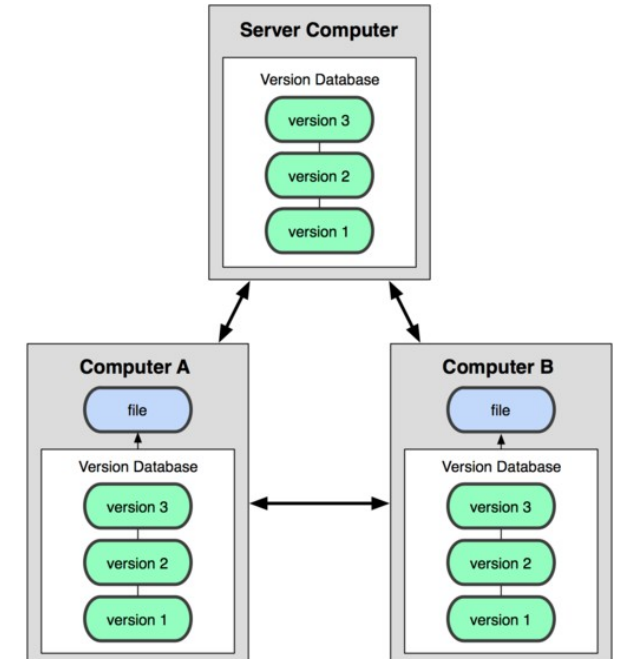
SCV Local (rcs)



SCV centralizado (cvs, subversion)



SCV distribuido (git, mercurial)





Gestión de código fuente con Git

- Ideado por Linus Torvalds para el desarrollo del núcleo de Linux
- Características principales
 - SCV distribuido de código abierto
 - Existen clientes gráficos para la mayoría de plataformas
 - Existen plugins para muchos IDEs (Eclipse, NetBeans, ...)
- Existen repositorios en Internet para alojar proyectos
 - GitHub (<http://github.com>)
 - Bitbucket (<https://bitbucket.org/>)



UNIVERSIDAD DE MÁLAGA



accenture technology

Sitio web de Git

 **git** --distributed-even-if-your-workflow-isnt

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.



<http://git-scm.com/> is easy to learn and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient staging areas, and **multiple workflows**.

 **Learn Git in your browser for free with Try Git.**



About

The advantages of Git compared to other source control systems.



Documentation

Command reference pages, Pro Git book content, videos and other material.



Downloads

GUI clients and binary releases for all major platforms.



Community

Get involved! Bug reporting, mailing list, chat, development and more.



Pro Git by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).



Windows GUIs



Tarballs



Mac Build



Source Code



Gestión de código fuente con Git

- GitBash es una consola básica que procesa líneas de órdenes Unix
- Conceptos básicos de Unix

Orden	Operación
<code>mkdir</code>	Crear directorio
<code>cd</code>	Cambiar directorio
<code>ls -la</code>	Listar directorio, modo extendido
<code>touch</code>	Crear un fichero
<code>cat</code>	Mostrar fichero por pantalla
<code>git</code>	Las órdenes para git comienzan por git



Gestión de código fuente con Git

- **Descargar e inicializar proyectos**
 - `clone`, `init`
- **Operaciones básicas**
 - `add`, `status`, `diff`, `commit`, `reset`, `mv`, `rm`
- **Ramas (branches) y fusiones (merges)**
 - `branch`, `checkout`, `merge`, `log`, `tab`
- **Compartir y actualizar proyectos**
 - `fetch`, `pull`, `push`, `remote`
- **Inspección y comparación**
 - `diff`, `log`



Creación de un repositorio local

- git init
- git config user.name
- git config user.email

```
projectoGit — bash — 80x24
berlin:git antelverde$ mkdir proyectoGit
berlin:git antelverde$ cd proyectoGit/
berlin:proyectoGit antelverde$ git init
Initialized empty Git repository in /Users/antelverde/Softw/git/proyectoGit/.git
/
berlin:proyectoGit antelverde$ git config user.name "martinav"
berlin:proyectoGit antelverde$ git config user.email "martinav@correo.com"
berlin:proyectoGit antelverde$ git status
# On branch master
#
# Initial commit
#
nothing to commit (create/copy files and use "git add" to track)
berlin:proyectoGit antelverde$
```

- git config --list



Creación de un repositorio local

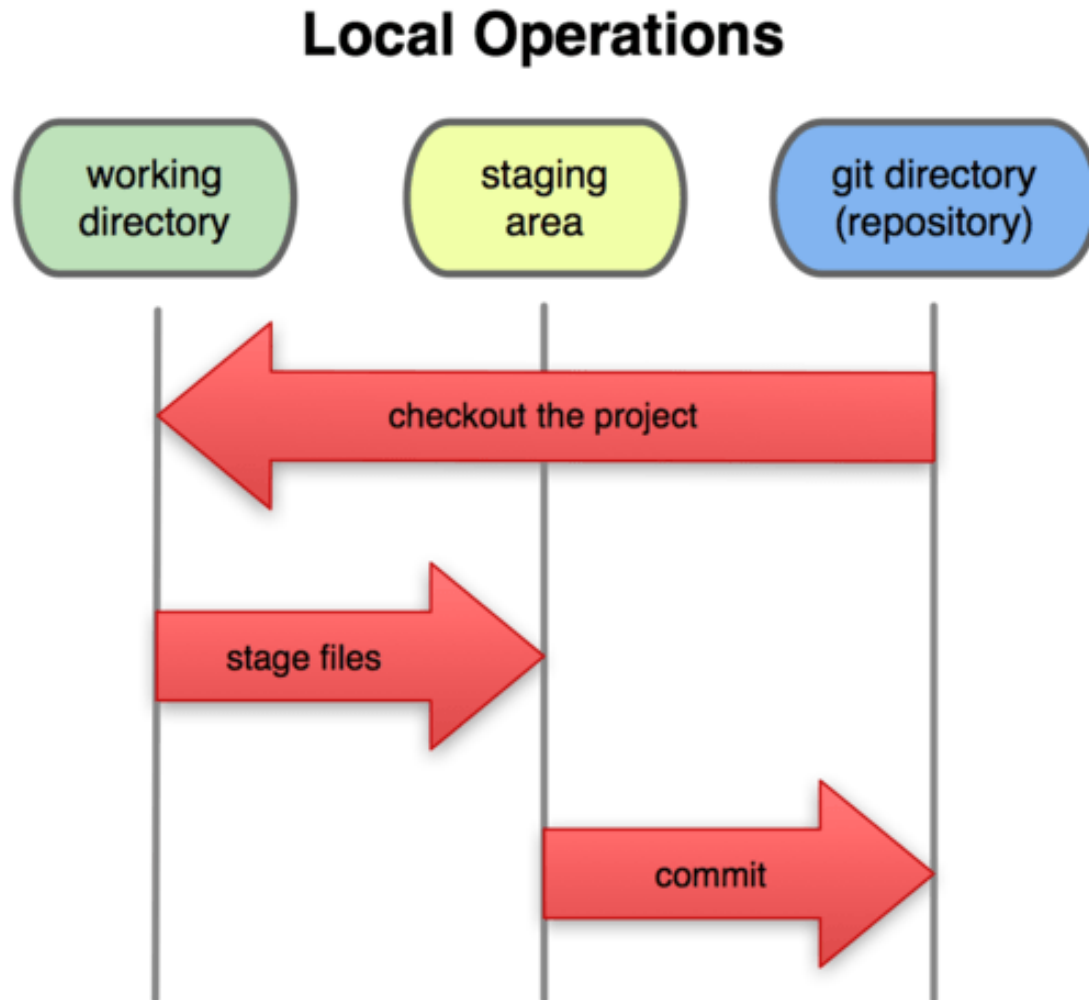
combinación de `ls -l` (para mostrar la lista en formato extendido) y `ls -a` (para no ignorar las entradas que comienzan con `.`)

```
bertoa@BER.../proyectoGit (master)
$ ls -la
total 6
drwxr-xr-x  1 bertoa  Administ   0 Feb 28 19:05 .
drwxr-xr-x 38 bertoa  Administ 8192 Feb 28 19:00 ..
drwxr-xr-x  9 bertoa  Administ 4096 Feb 28 19:09 .git
```



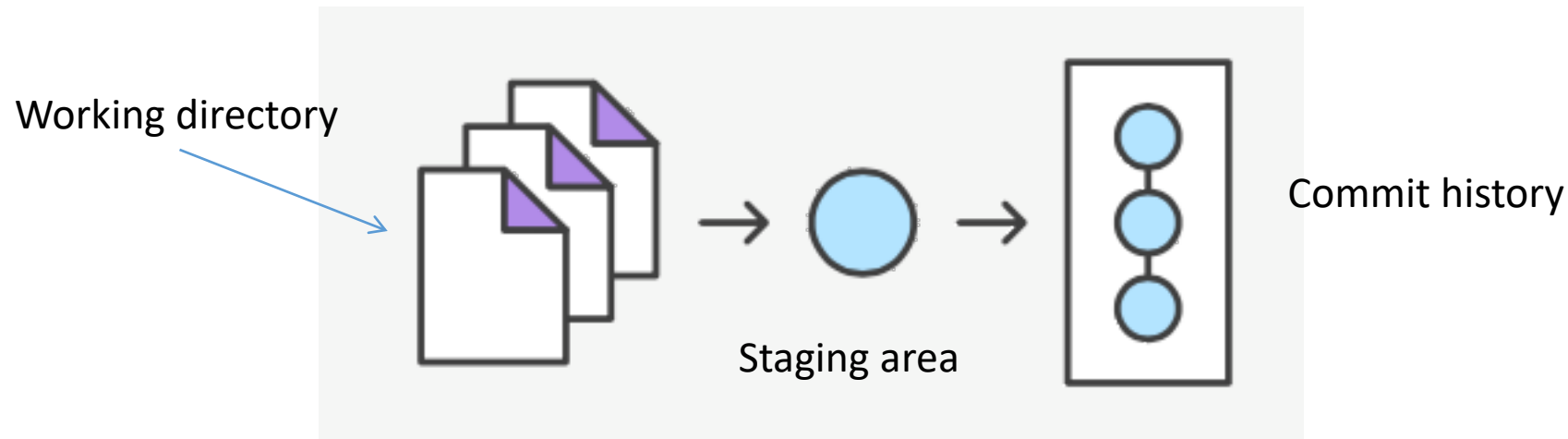
Repositorio git en modo local

- Las tres áreas



Staging area

- Es un lugar lógico donde los cambios residen antes de que son consolidados
- Sirve para tener un control más preciso de lo que los desarrolladores quieren que sea parte de la siguiente consolidación (y de lo que no)
- En CVS o SVN todos los cambios son consolidados sin lugar a opción





Añadir ficheros a la *staging* area

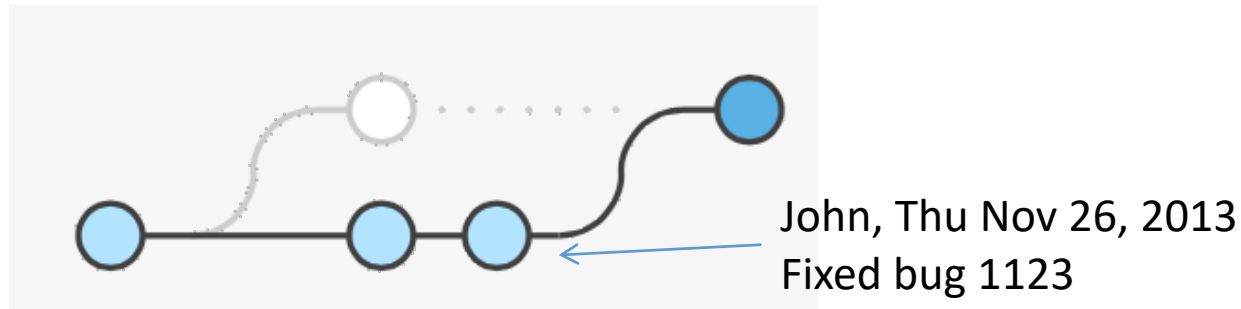
- `git add <archivos>`

```
projectoGit — bash — 80x22
berlin:proyectoGit antelverde$ vi Hola.c
berlin:proyectoGit antelverde$ cat Hola.c
#include <stdio.h>
int main() {
    printf("Hola, mundo\n") ;
}
berlin:proyectoGit antelverde$ gcc Hola.c -o Hola
berlin:proyectoGit antelverde$ ./Hola
Hola, mundo
berlin:proyectoGit antelverde$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   Hola
#   Hola.c
nothing added to commit but untracked files present (use "git add" to track)
berlin:proyectoGit antelverde$ git add Hola.c
berlin:proyectoGit antelverde$
```



Consolidar cambios en el repositorio

- El **commit** es uno de los conceptos más importantes en Git
- En pocas palabras, un commit es una instantánea del código fuente con sus propiedades (desarrollador, fecha, comentarios, etc.)





Consolidar cambios en el repositorio

- `git commit -m "message"`

```
projectoGit — bash — 80x22
berlin:proyectoGit antelverde$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#   new file:   Hola.c
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       Hola
berlin:proyectoGit antelverde$ git commit -m "Hola.c incluido"
[master (root-commit) 75a70de] Hola.c incluido
 1 files changed, 4 insertions(+), 0 deletions(-)
 create mode 100644 Hola.c
berlin:proyectoGit antelverde$
```



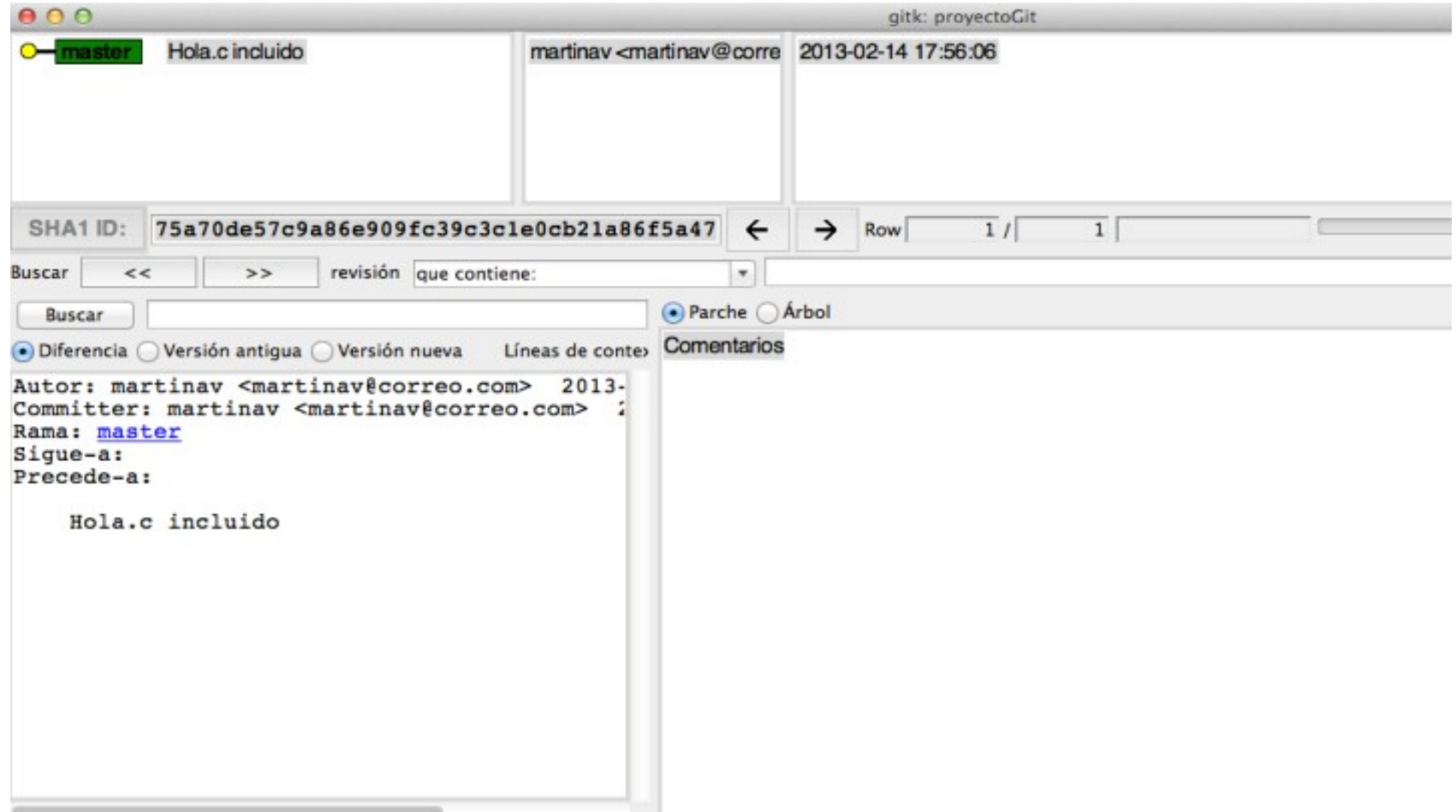
Ejercicio

- Crear un programa Java sencillo (HelloWorld.java) o bien elegir uno que ya hayáis desarrollado con anterioridad y subirlo al repositorio local que hemos creado
 - Los ficheros .class deben ser omitidos
- Buscar información acerca de .gitignore
 - ¿Para qué sirve?
 - ¿Qué formato tiene?



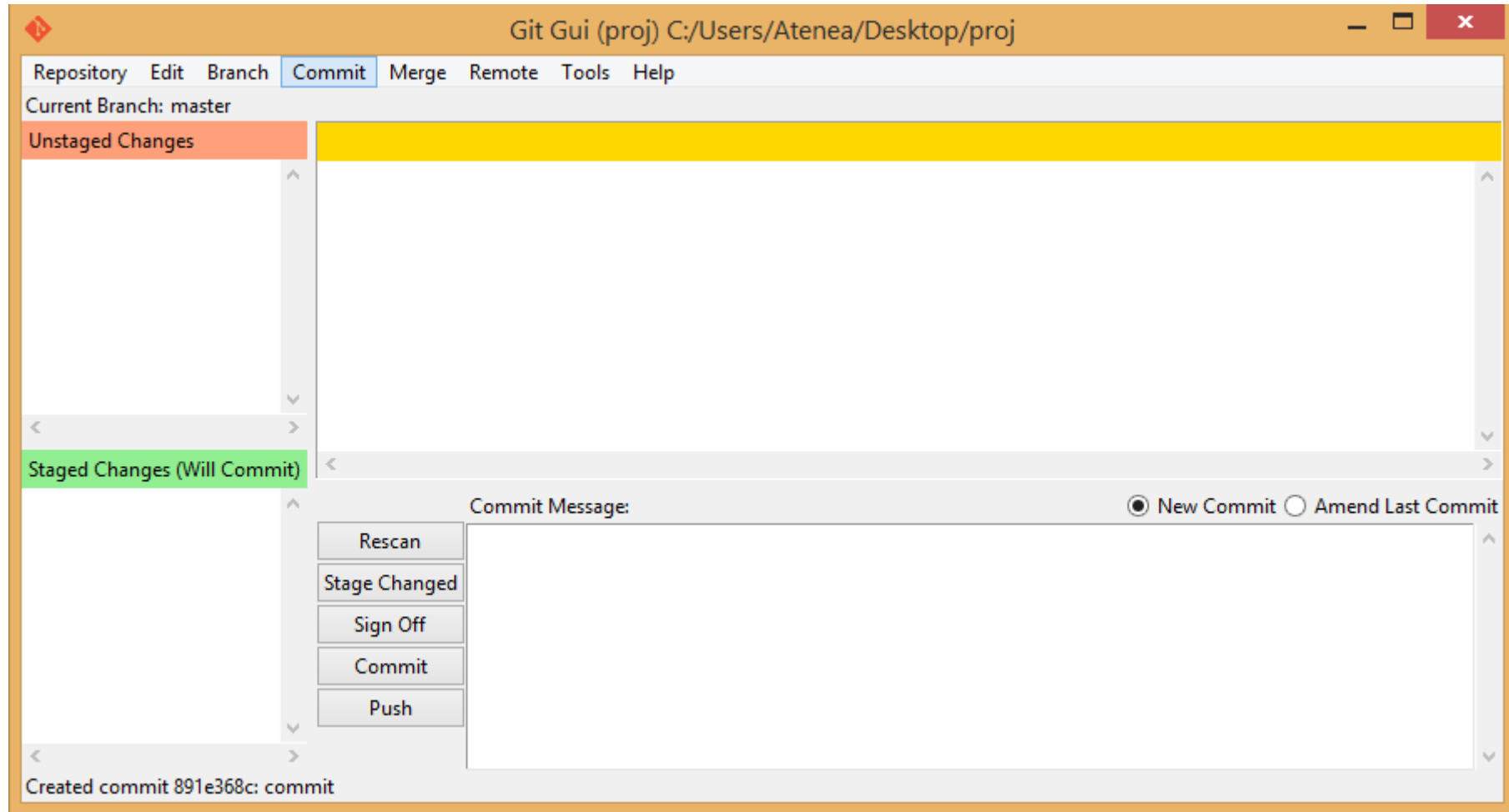
Interfaz gráfica gitk

- gitk --all &





Interfaz gráfica Git GUI





Modificación de un fichero y actualización

```
projectoGit — bash — 80x24
berlin:proyectoGit antelverde$ vi Hola.c
berlin:proyectoGit antelverde$ cat Hola.c
#include <stdio.h>
int main() {
    /* Programa C que imprime un mensaje por pantalla */
    printf("Hola, mundo\n") ;
}
berlin:proyectoGit antelverde$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#   modified:   Hola.c
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       Hola
no changes added to commit (use "git add" and/or "git commit -a")
berlin:proyectoGit antelverde$ git commit -a -m "Actualizado Hola.c"
[master 6fd67e4] Actualizado Hola.c
 1 files changed, 1 insertions(+), 0 deletions(-)
berlin:proyectoGit antelverde$
```



Ejercicio

- Refrescar gitk después de la modificación del fichero anterior
- Añadir un nuevo fichero y ver qué ha ocurrido en Git GUI.
- Añadirlo (add) y volver a observar los cambios en Git GUI.
- Hacer el commit del archivo desde Git GUI.



Viendo los cambios

- `git diff <archivo>`

```
MINGW32:/c/Users/Atenea/Desktop/proj

Atenea@ATENEAPC ~/Desktop/proj (master)
$ git diff hello.c

@@ -1,7 +1,6 @@
#include <stdio.h>
int main()
{
- // printf() displays the string inside quotation
- printf("Hello, World!");
+ printf("Hello, World!!!!");
return 0;
}
\ No newline at end of file

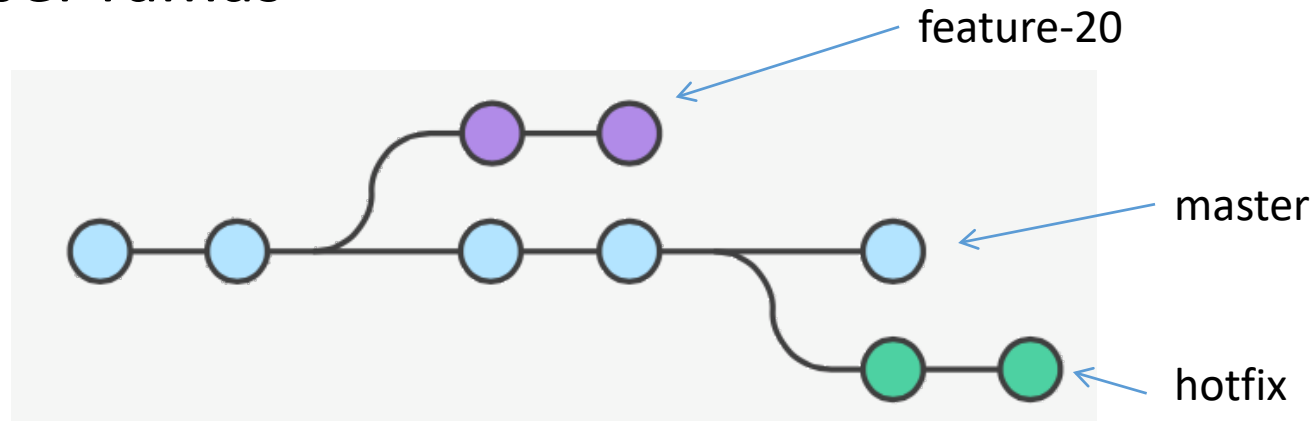
Atenea@ATENEAPC ~/Desktop/proj (master)
$
```

Línea borrada

Línea modificada = borrada + insertada

Ramas

- En Git los commits no tienen porque seguir una secuencia lineal sino que puede haber ramas



- A cada rama se le asigna un nombre para distinguirla de la demás
- Aunque desde un punto de vista técnico no hay ninguna diferencia, se considera que hay una rama por defecto la cual se llama **master**



Ramas

- Al igual que una rama se puede diversificar, también se puede fusionar con otra
- El comando para consultar las ramas es: git branch
- La rama activa se muestra con un asterisco

```
projectoGit — bash — 80x23
berlin:projectoGit antelverde$ git branch
* master
berlin:projectoGit antelverde$
```



Ramas

- Creación de una rama: `git branch <nombre rama>`
- Cambiar de una rama a otra: `git checkout <nombre de la rama>`

```
$ git branch
* master
bertoa@BERTOA-PC ~/Documents/practicaGIT (master)
$ git branch testing
bertoa@BERTOA-PC ~/Documents/practicaGIT (master)
$ git branch
* master
  testing
bertoa@BERTOA-PC ~/Documents/practicaGIT (master)
$ git checkout testing
Switched to branch 'testing'
bertoa@BERTOA-PC ~/Documents/practicaGIT (testing)
$ git branch
  master
* testing
bertoa@BERTOA-PC ~/Documents/practicaGIT (testing)
$
```




Ejercicio

- Apartado 1
 - Escogemos un archivo que tengamos en nuestro repositorio
 - Creamos una nueva rama (llamada testing) y nos cambiamos a ella
 - Modificamos el archivo escogido
 - Consolidamos los cambios en esa rama
 - Volvemos a la rama master y vemos que el archivo en esa rama se mantiene en su estado original

- Apartado 2
 - Creamos un nuevo archivo en la rama testing (lo consolidamos)
 - Creamos un nuevo archivo en la rama master (lo consolidamos)

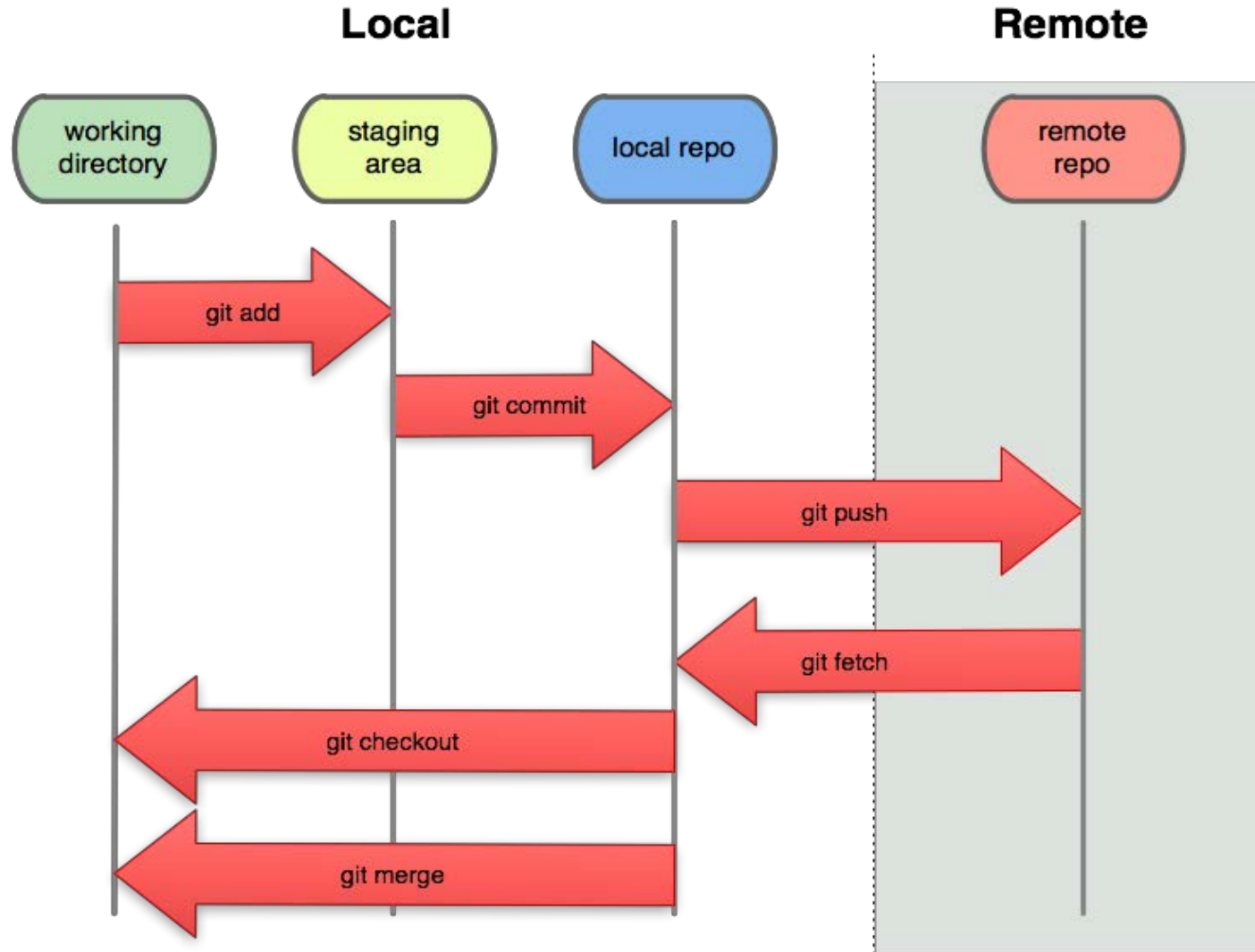


Ejercicio

- Apartado 3
 - Desde la rama master, fusiona ambas ramas usando:
git merge <nombre rama>
 - Usa gitk para ver todos los cambios realizados
 - Comprueba los archivos para ver qué cambios prevalecen



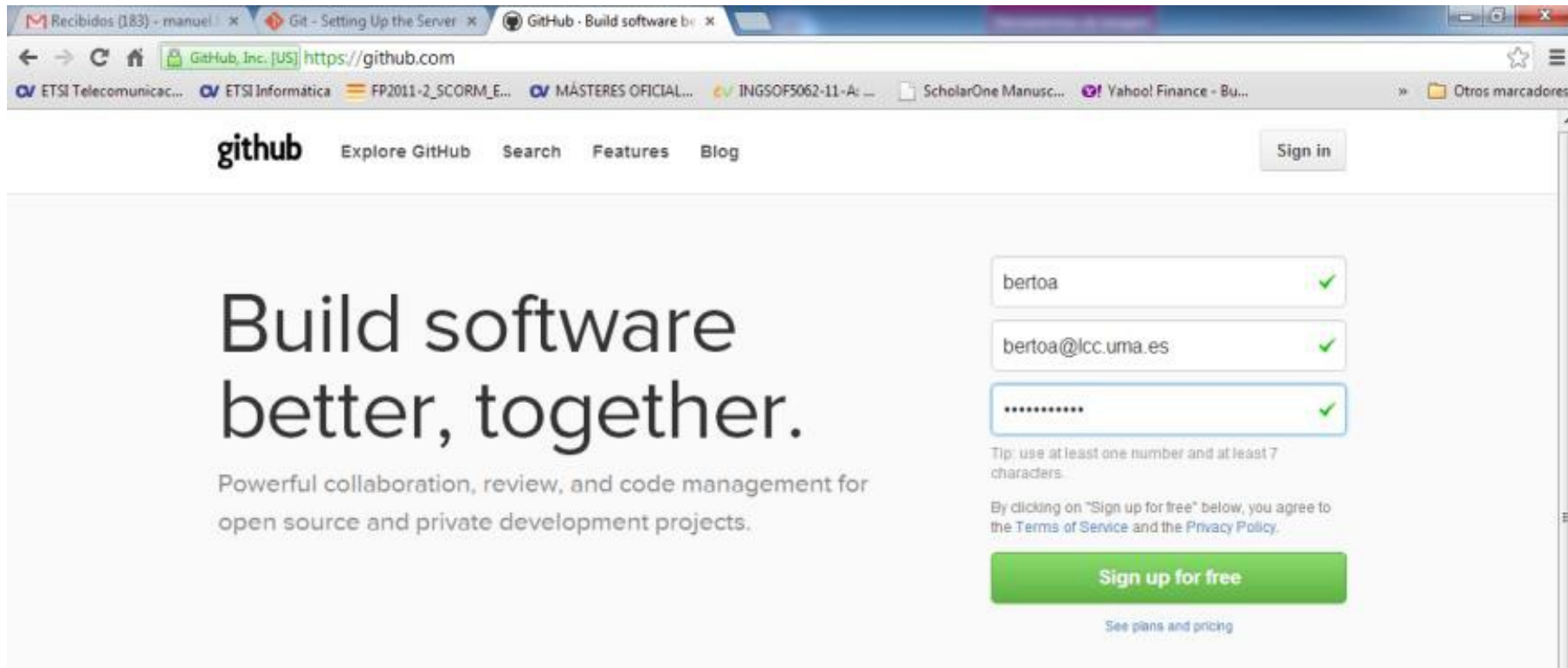
Git en modo remoto





Ejercicio

- Crear una cuenta en GitHub





Ejercicio

- Crear un repositorio en GitHub

Search or type a command

Explore Gist Blog Help

bertoa

Owner: bertoa / Repository name: []

Great repository names are short and memorable. Need inspiration? How about **drunken-hipster**.

Description (optional)

Public
Anyone can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with a README
This will allow you to `git clone` the repository immediately.

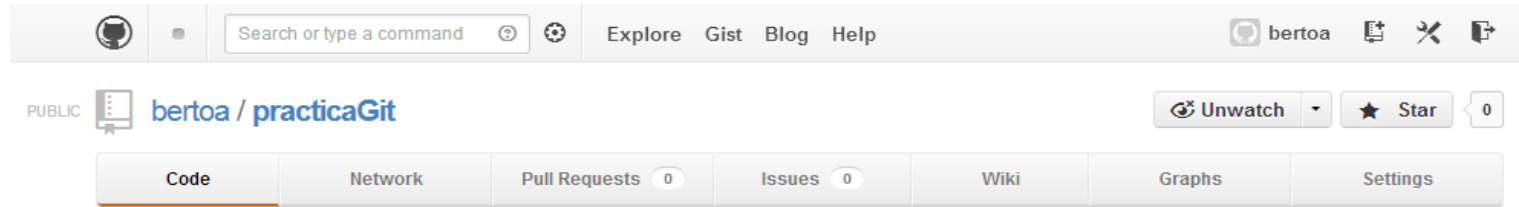
Add .gitignore: **None**

Create repository



Ejercicio

- Subir el repositorio local al remoto



Quick setup — if you've done this kind of thing before

Setup in Windows or

HTTP SSH `https://github.com/bertoa/practicaGit.git`

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

Create a new repository on the command line

```
touch README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/bertoa/practicaGit.git
git push -u origin master
```

Push an existing repository from the command line

```
git remote add origin https://github.com/bertoa/practicaGit.git
git push -u origin master
```



Ejercicio

- Subir el repositorio local al remoto

```
git remote add origin https://github.com/bertoa/practicaGit.git  
git push -u origin master
```

```
Git Bash  
bertoa@BERTOA-PC ~/Documents/practicaGIT (master)  
$ git remote add origin https://github.com/bertoa/practicaGit.git  
bertoa@BERTOA-PC ~/Documents/practicaGIT (master)  
$ git push -u origin master  
Username for 'https://github.com': bertoa  
Password for 'https://bertoagithub.com':  
Counting objects: 22, done.  
Delta compression using up to 4 threads.  
Compressing objects: 100% (21/21), done.  
Writing objects: 100% (22/22), 2.56 KiB, done.  
Total 22 (delta 7), reused 0 (delta 0)  
To https://github.com/bertoa/practicaGit.git  
 * [new branch]      master -> master  
Branch master set up to track remote branch master from origin.  
bertoa@BERTOA-PC ~/Documents/practicaGIT (master)  
$
```

- Clonar un repositorio remoto: `git clone <url del repositorio>`



Ejercicio por parejas

- Paso 1:
 - Alumno1:
 - Dar permisos al Alumno2 para poder editar en el repositorio
 - Crear un directorio llamado practicaGit y dentro un fichero llamado GitAux.java que contenga el siguiente método y hacer commit

```
public class GitAux {  
    public void metodoAlumno1(){  
        System.out.println("metodo 1");  
    }  
  
    public void metodoComunitario(){  
        // Comentario incluido para pruebas  
        System.out.println("Aqui escribimos todos");  
    }  
}
```




Ejercicio por parejas

- Paso 2: Simultáneamente

- Alumno 1:

- Añadir la siguiente línea al metodoAlumno1():

```
System.out.println("modificado por alum 1"); // insertado por alum1
```

- y la siguiente línea al metodoComunitario():

```
System.out.println("alumn 1 inserta esta sentencia");
```

- Alumno 2:

- Clonar el repositorio del Alumno1
 - Editar el archivo GitAux tal que añada un método metodoAlumno2() y modifique el método comunitario tal que así:

```
public void metodoAlumno2(){  
    System.out.println("metodo 2");  
}
```

```
public void metodoComunitario(){  
    // Comentario incluido para pruebas  
    System.out.println("Aqui escribimos todos");  
    System.out.println("alumno 2"); //modif por alumno 2  
}
```



Ejercicio por parejas

- Paso 3:
 - Alumno 1:
 - Hace commit y sube al repositorio sus cambios (push)
- Paso 4:
 - Alumno 2:
 - Hace commit y sube al repositorio sus cambios (push)
 - El push fallará! ¿Por qué?



Ejercicio

- Paso 5:

- Alumno 2:

- Hacer pull (hará automáticamente un merge)
 - Resolver conflictos!

No hay conflicto
Zonas diferentes

Hay conflicto
Misma zona

```
GitAux.java (~\Documents\prbGITHUB\practicaGit) - VIM
package prgitejemplo;

/**
 * @author MFB
 */
public class GitAux {

    public void metodoAlumno1(){
        System.out.println("metodo 1");
        System.out.println("modificado por alum 1"); // insertado por alum1
    }

    public void metodoAlumno2(){
        System.out.println("metodo 2");
        System.out.println("insertado por alumno2"); // modif alumno 2
    }

    public void metodoComunitario(){
        // Comentario incluido para pruebas
        System.out.println("Aquí escribimos todos");
    }
}

<<<<<<< HEAD
// modificacaciones alumno 1
System.out.println("Alumno 1 inserta esta sentencia");
=====
System.out.println("alumno 2"); // insertado por alumno 2
>>>>>>> 9f12134c052af93ff4059fdbdd75f6fb3aadea35
}
public void testGitAux(){
    // metodo rama testing
    System.out.println("Metodo de testing");
}
}

~\Documents\prbGITHUB\practicaGit\GitAux.java [dos] (16:46 02/03/2013) 37,1 Bot
```



Conflictos

- Si dos usuarios modifican un mismo fichero, al hacer un merge se pueden producir conflictos
- ¿Cómo se resuelven?
 - En el fichero aparece una anotación de las instrucciones en conflicto
 - Hay que modificar el fichero manualmente para corregir el problema
 - El fichero resultante se trata como un fichero modificado más:
 - hay que hacer `add` y `commit`



Índice

- Gestión de código fuente con Git
- Construcción de código con Maven
- Integración continua en Jenkins



¿Qué es Maven?

- Una herramienta para automatizar la construcción de software
- Parecida a **make** o **Ant**, pero mucho más potente
- Uno de los mayores problemas de las herramientas anteriores es la gestión de dependencias
 - ... que lleva al llamado **“Jar Hell”**
- Maven soluciona (o reduce) este problema
- <http://maven.apache.org>



Instalación de Maven

- Fichero README.txt

...

- 1) Unpack the archive where you would like to store the binaries, eg:

Unix-based operating systems (Linux, Solaris and Mac OS X)

```
tar zxvf apache-maven-3.x.y.tar.gz
```

Windows

```
unzip apache-maven-3.x.y.zip
```

- 2) A directory called "apache-maven-3.x.y" will be created.

- 3) Add the bin directory to your PATH, eg:

```
set PATH="c:\program files\apache-maven-3.x.y\bin";%PATH%
```

- 4) Make sure JAVA_HOME is set to the location of your JDK

- 5) Run "mvn --version" to verify that it is correctly installed.

...



Fichero POM

- El fichero POM es un XML donde se encuentra toda la información relacionada con los pasos de construcción, incluidas las dependencias

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.2</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

- Cuando se especifica una dependencia en el POM, Maven la descarga automáticamente y la resuelve
- Todas las dependencias se guardan en un repositorio local por lo que solo las descarga una vez



Ciclo de vida en Maven

- Las tareas Maven son ejecutadas en el orden dado por un ciclo de vida

Ciclo de vida por defecto

- validate
- initialize
- generate-sources
- process-sources
- generate-resources
- process-resources
- compile
- process-classes
- generate-test-sources
- process-test-sources
- generate-test-resources
- process-test-resources
- test-compile

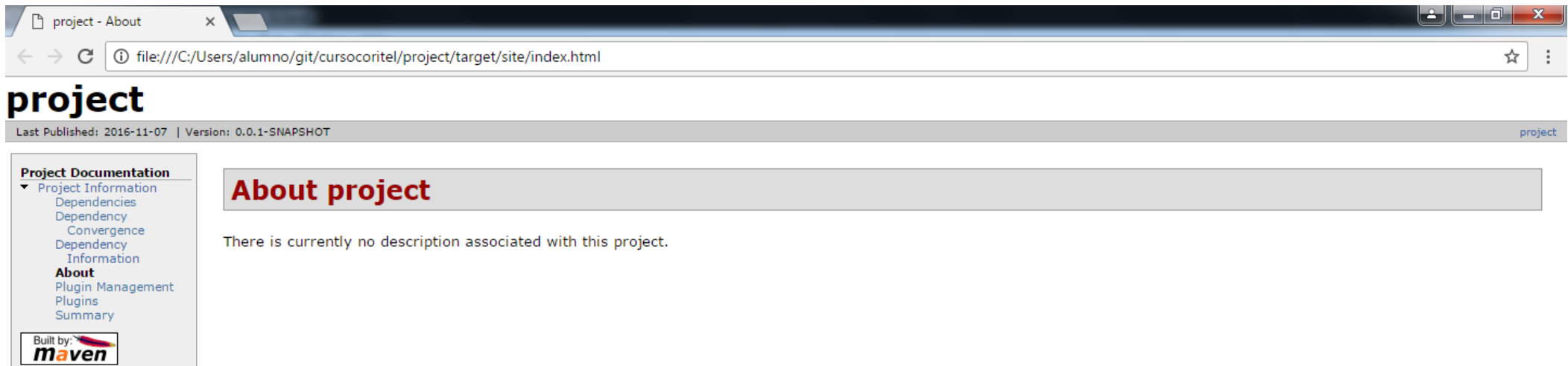
- process-test-classes
- test
- prepare-package
- package
- pre-integration-test
- integration-test
- post-integration-test
- verify
- install
- deploy

Cuando se le proporciona una fase a Maven, Maven ejecuta todas las fases desde la primera hasta que alcanza la misma



Documentación

- Con la orden:
`mvn site`
maven crea automáticamente un sitio web para el proyecto





Maven

- Para ampliar conocimientos:

https://www.tutorialspoint.com/maven/maven_project_documents.htm



Índice

- Gestión de código fuente con Git
- Construcción de código con Maven
- Integración continua en Jenkins



Jenkins

- Jenkins es una herramienta de la integración continua
 - Pretende realizar *integraciones automáticas* de un proyecto lo más a menudo posible para así poder detectar fallos cuanto antes.
 - Entendemos por integración la compilación y ejecución de pruebas de todo un proyecto

- Open source

- Escrita en Java



Jenkins

- Pasos para instalar y usar Jenkins:
 - Ejecutar en un servidor (p.ej., Tomcat, Wildfly) o en modo standalone
 - No requiere base de datos
 - Configurar el servidor
 - Añadir usuarios
 - Añadir los credenciales
 - Instalar plugins
 - Añadir tareas



Jenkins

- Pasos para instalar y usar Jenkins:
 - Ejecutar en un servidor (p.ej., Tomcat, Wildfly) o en modo standalone
 - No requiere base de datos
 - Configurar el servidor
 - Añadir usuarios
 - Añadir los credenciales
 - Instalar plugins
 - Añadir tareas

En nuestro caso...

- 1) Lanzar el servidor Wildfly ejecutando `/bin/standalone.bat`
- 2) Acceder desde el navegador a `localhost:8080` y ver que se está ejecutando
- 3) Ejecutar `/bin/add-user.bat` para añadir un usuario que pueda acceder a la consola de administración
- 4) En la pestaña Deployments, añadir `jenkins.war`
- 5) Comprobar que Jenkins se está ejecutando (`localhost:8080/Jenkins`)



Jenkins

UNIVERSIDAD DE MÁLAGA



accenturetechnology

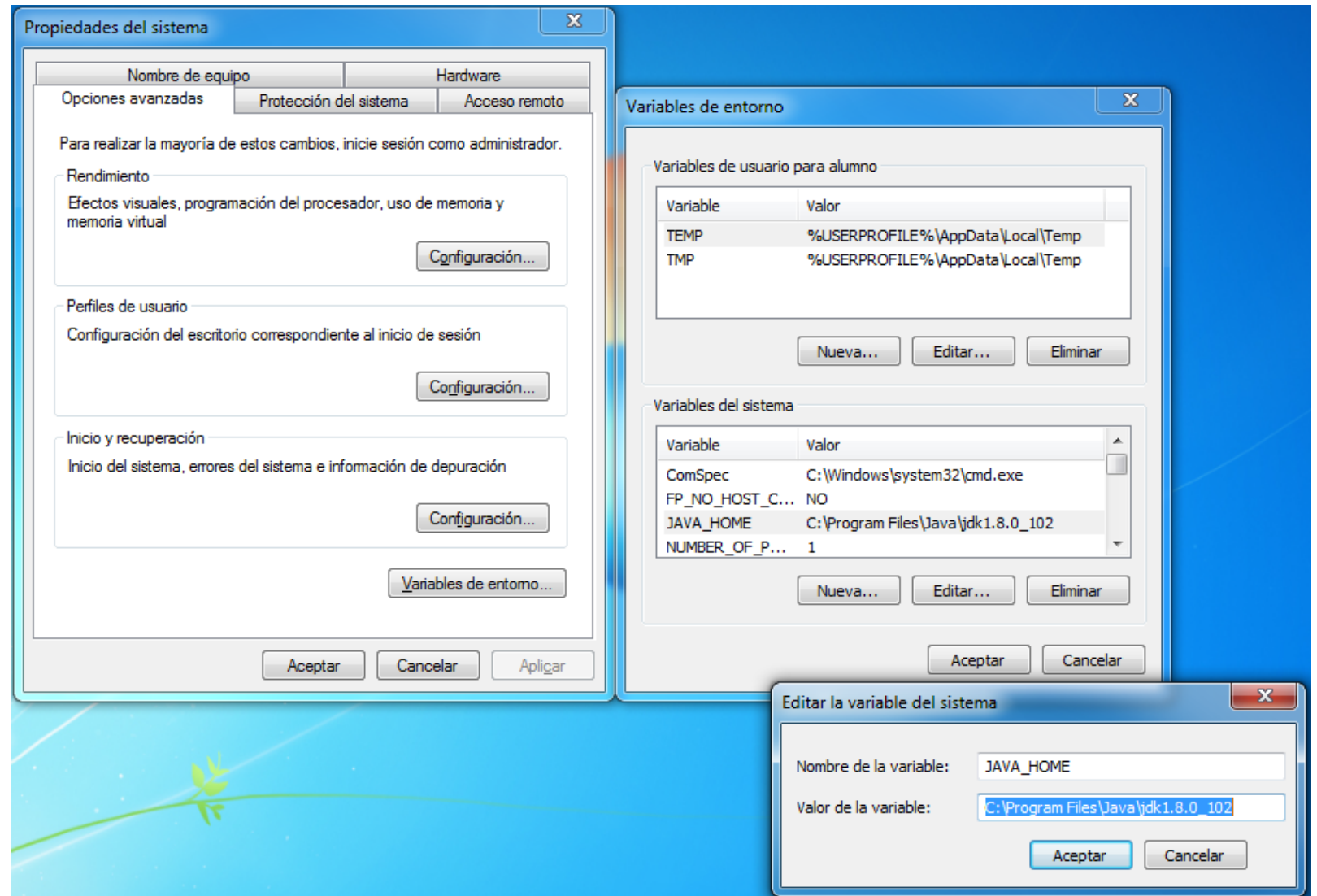
The screenshot shows a Windows desktop environment. In the background, a File Explorer window is open to the directory `wildfly-10.1.0.Final\bin`, listing files such as `launcher`, `product.conf`, and `standalone`. In the foreground, a Command Prompt window displays the output of a `cmd.exe` process, showing logs for WildFly 10 startup. The logs include messages about passivation configuration, cache eviction, and the successful start of various services. A Mozilla Firefox browser window is also open, displaying the 'Welcome to WildFly 10' page at `localhost:8080`. The page features the WildFly logo and navigation links for documentation, quickstarts, and the administration console.


```
C:\Windows\system32\cmd.exe
What type of user do you wish to add?
a) Management User <mgmt-users.properties>
b) Application User <application-users.properties>
(a):
Enter the details of the new user to be added
Using realm 'ManagementRealm' as default realm
Username : loli
Password recommendations are listed below:
- The password should be different from the realm name and user name
- The password should not be one of the following:
n, administrator
- The password should contain at least 8 characters, 1 digit(s), 1 non-alphanumeric symbol
Password :
Re-enter Password :
What groups do you want this user to belong to? Use space to separate
list, or leave blank for none) [ ]:
About to add user 'loli' for realm 'ManagementRealm'
Is this correct yes/no? yes
Added user 'loli' to file 'C:\Users\alvarez\Documents\wildfly-10.1.0.Final\standalone\configuration\standalone-users.xml'
Added user 'loli' to file 'C:\Users\alvarez\Documents\wildfly-10.1.0.Final\domain\configuration\domain-users.xml'
Added user 'loli' with groups to file 'C:\Users\alvarez\Documents\wildfly-10.1.0.Final\standalone\configuration\standalone-users.xml'
Added user 'loli' with groups to file 'C:\Users\alvarez\Documents\wildfly-10.1.0.Final\domain\configuration\domain-users.xml'
Is this new user going to be used for remote access?
e.g. for a slave host controller connection for server to server EJB calls
yes/no? yes
To represent the user add the following to the configuration file:
ret value="YnUyZ3Ulbm8uMTIz" />
Presione una tecla para continuar . . .
```

WildFly Management console showing the WildFly homepage. The console displays the WildFly logo and navigation tabs: Home, Deployments, Configuration, Runtime, Access Control, Patching. The main content area shows the WildFly management interface with sections for Deployments, Configuration, Runtime, Access Control, and Patching. The console also shows the output of the 'add-user' command, indicating that the user 'loli' has been successfully added to the system.

Jenkins

- Si obtenemos un error debido a que la variable de entorno `JAVA_HOME` no existe deberemos añadirla





Jenkins

- **Pasos para instalar y usar Jenkins:**
 - Ejecutar en un servidor (p.ej., Tomcat, Wildfly) o en modo standalone
 - No requiere base de datos
 - Configurar el servidor
 - Añadir usuarios
 - Añadir los credenciales
 - **Instalar plugins**
 - **Añadir tareas**

- Introducimos la clave que nos pide,
- Instalamos los plugins y por último...
- Creamos nuestro usuario

The image displays two overlapping browser windows showing the Jenkins web interface. The background window is titled 'Jenkins [Jenkins]' and shows the 'Getting Started' page with the heading 'Unlock Jenkins'. The text on this page explains that a password has been written to a log file on the server. A code block shows the path: `C:\Users\alumno\.jenkins\secrets\initialAdminPassword`. Below this, there is a text input field labeled 'Administrator password'. The foreground window is titled 'SetupWizard [Jenkins]' and shows the 'Getting Started' page with the heading 'Create First Admin User'. It contains several form fields: 'Usuario:', 'Contraseña:', 'Confirma la contraseña:', 'Nombre completo:', and 'Dirección de email:'. At the bottom of this window, there is a 'Save and Finish' button and the text 'Continue as admin'.



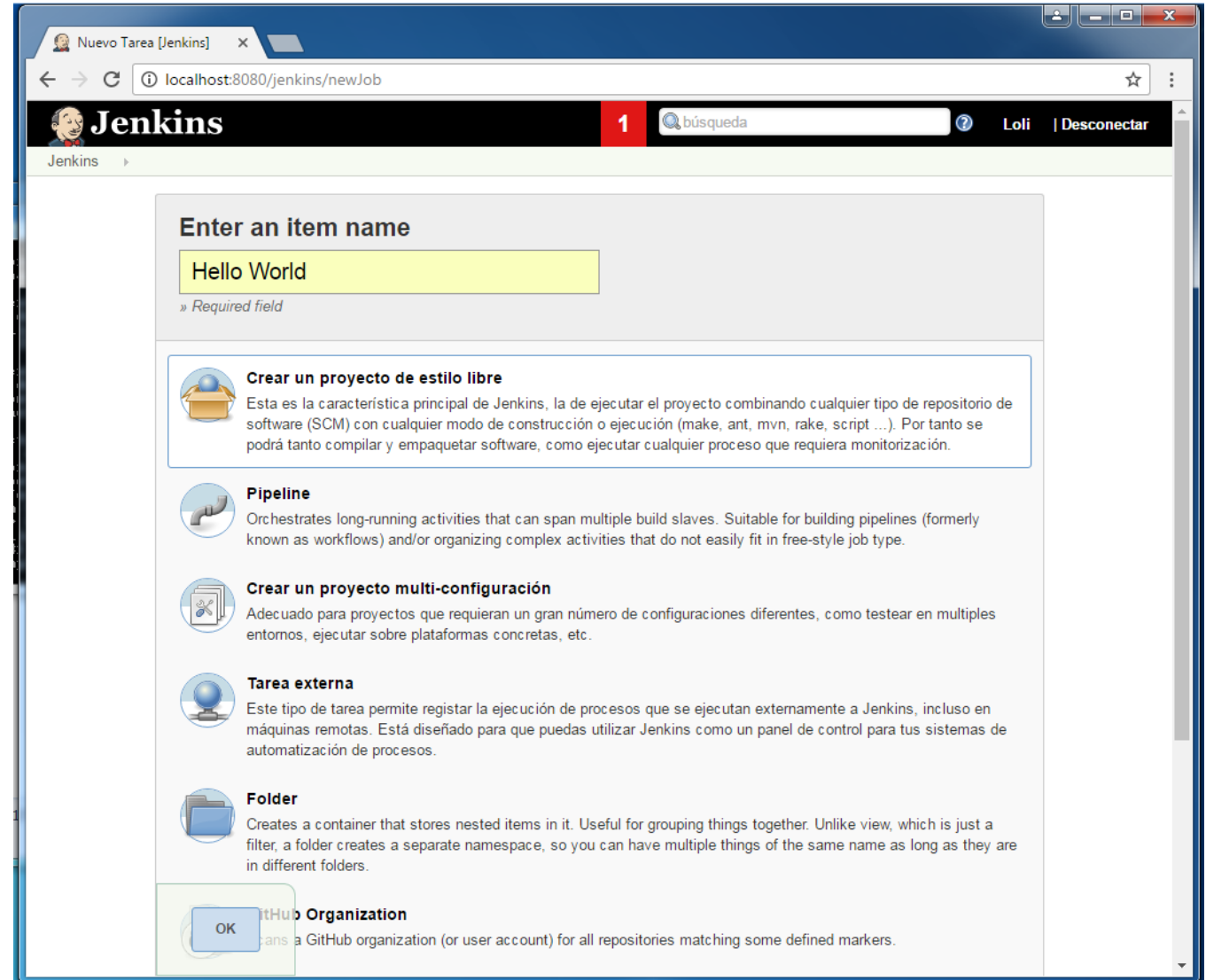
Jenkins

The screenshot shows the Jenkins web interface in a browser window. The browser tabs include 'WildFly Management' and 'Panel de control [Jenkins]'. The address bar shows 'localhost:8080/jenkins/'. The page header features the Jenkins logo, a search bar, and the user name 'Loli' with a 'Desconectar' link. A red notification badge with the number '1' is visible. The main content area displays a welcome message: '¡Bienvenido a Jenkins!' followed by a prompt: 'Por favor, **crea una nueva tarea** para empezar.' The left sidebar contains navigation links: 'Nueva Tarea', 'Personas', 'Historial de trabajos', 'Administrar Jenkins', 'Mis vistas', and 'Credentials'. Below these are two expandable sections: 'Trabajos en la cola' (showing 'No hay trabajos en la cola') and 'Estado del ejecutor de construcciones' (showing two 'Inactivo' entries). The footer indicates the page was generated on 04-nov-2016 at 14:59:26 CET, with links for 'REST API' and 'Jenkins ver. 2.25'.



Jenkins

- Creamos un proyecto de estilo libre nuevo...





Jenkins

- ... y lo configuramos para que coja el código fuente del repositorio Git:
<https://github.com/lolybc88/cursocoritel.git>
- Requiere que Git esté instalado en el sistema

The screenshot shows the Jenkins configuration page titled "Configurar el origen del código fuente". It has two radio buttons: "Ninguno" (unselected) and "Git" (selected). Under "Repositories", there is a text input for "Repository URL" containing "https://github.com/lolybc88/cursocoritel.git", a dropdown for "Credentials" set to "- none -", and an "Add" button. Below these are "Avanzado..." and "Add Repository" buttons. Under "Branches to build", there is a text input for "Branch Specifier (blank for 'any')" containing "*/master" and an "Add Branch" button. A red "X" icon is visible above the branch specifier input.

Si no encuentra dónde Git está instalado en el sistema:

Manage Jenkins > Configure System settings y en la sección Git, cambiar: **Path to Git executable** para que apunte a la ubicación del fichero git.exe



Jenkins

- ... y para que se ejecute a cada hora (13:00, 14:00, etc.)

Disparadores de ejecuciones

- Lanzar ejecuciones remotas (ejem: desde 'scripts')
- Build after other projects are built
- Build when a change is pushed to GitHub
- Consultar repositorio (SCM)
- Ejecutar periódicamente

Programador: 0 * * * *



Jenkins

The screenshot shows the Jenkins dashboard for a project named "Hello World". The left sidebar contains navigation options: "Volver al Panel de Control", "Estado Actual", "Cambios", "Zona de Trabajo", "Construir ahora", "Borrar Proyecto", "Configurar", and "Move". The main content area is titled "Proyecto Hello World" and includes a search bar, a "Desactivar el Proyecto" button, and a "Historia de tareas" section. The "Historia de tareas" section shows a table with columns for build number and time. An orange arrow points to the "Historia de tareas" section, and another orange arrow points to the "Estado Actual" link in the sidebar.

Build	Time
#1	04-nov-2016 16:29

The screenshot shows the Jenkins console output for the "Hello World" project. The console output displays the following commands and their results:

```
Lanzada por el usuario Loli
Ejecutando en el espacio de trabajo C:\Users\loli\AppData\Local\Microsoft\Windows\Workspaces\Hello World
> git.exe rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git.exe config remote.origin.url https://github.com/lolybc88/cursocoritel.git # timeout=10
Fetching upstream changes from https://github.com/lolybc88/cursocoritel.git
> git.exe --version # timeout=10
> git.exe fetch --tags --progress https://github.com/lolybc88/cursocoritel.git +refs/heads/*:refs/remotes/origin/*
> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
> git.exe rev-parse "refs/remotes/origin/origin/master^{commit}" # timeout=10
Checking out Revision 6ecdfa178237bb271d83288d1d66ec9cc0a6822b (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f 6ecdfa178237bb271d83288d1d66ec9cc0a6822b
First time build. Skipping changelog.
Finished: SUCCESS
```

An orange arrow points to the "Salida de consola" section header.



Git y Jenkins

Configurar el origen del código fuente

Ninguno
 Git

Repositories

Repository URL

Credentials

Branches to build

Branch Specifier (blank for 'any')

Navegador del repositorio

Additional Behaviours

Merge before build

Name of repository

Branch to merge to

Merge strategy

Fast-forward mode

Acciones para ejecutar después.

Git Publisher

Push Only If Build Succeeds

Merge Results

If pre-build merging is configured, push the result back to the origin



SonarQube y Jenkins

Instalación...

- Pre-requisito: SonarQube (y su base de datos) debe estar instalado
- Usando el gestor de plugins de Jenkins instalamos el plugin de SonarQube

Configuración del servidor...

- Administrar Jenkins > Configuración del Sistema > SonarQube servers

SonarQube servers

Environment variables

Instalaciones de SonarQube

Enable injection of SonarQube server configuration as build environment variables
If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Name	my SonarQube
URL del servidor	http://localhost:9000 <small>Por defecto es http://localhost:9000</small>
Server version	5.3 or higher
Server authentication token	
SonarQube account login	
SonarQube account password	

Configuration fields depend on the SonarQube server version.

SonarQube authentication token. Mandatory when anonymous access is disabled.

SonarQube account used to perform analysis. Mandatory when anonymous access is disabled. No longer used since SonarQube 5.3.

SonarQube account used to perform analysis. Mandatory when anonymous access is disabled. No longer used since SonarQube 5.3.

Avanzado...

Delete SonarQube



SonarQube y Jenkins

Añadiendo el scanner...

- Administrar Jenkins > Global tool configuration

The screenshot shows the Jenkins 'Global Tool Configuration' page for 'SonarQube Scanner'. It features a table with one entry: 'SonarQube Scanner' with the name 'my sonarqube scanner'. The 'Install automatically' checkbox is checked. Below the table, there are buttons for 'Añadir un instalador', 'Añadir SonarQube Scanner', 'Borrar un instalador', and 'Borrar SonarQube Scanner'. The version 'SonarQube Scanner 2.8' is selected in the 'Versión' dropdown.

- Podemos escoger una version instalada (deseleccionando '*Install automatically*') o permitir a Jenkins realizar la instalación desde una ubicación remota (seleccionado '*Install automatically*')



SonarQube y Jenkins

Una vez instalado...

- En el proyecto que se quiera aplicar: Configurar > Ejecutar > Añadir un Nuevo paso > Execute SonarQube Scanner
- Podemos o bien seleccionar un fichero .properties o bien detallar las properties en el campo Analysis properties.

The screenshot shows the Jenkins configuration interface for the 'Execute SonarQube Scanner' step. The title is 'Ejecutar' and the step name is 'Execute SonarQube Scanner'. The configuration fields are:

- Task to run:** An empty text input field.
- JDK:** A dropdown menu set to '(Inherit From Job)'. Below it, the text 'JDK to be used for this SonarQube analysis' is displayed.
- Path to project properties:** An empty text input field.
- Analysis properties:** A text area containing the following properties:

```
sonar.projectKey=my:helloworld  
sonar.projectName=hello world  
sonar.projectVersion=1.0  
  
sonar.sources=.
```
- Additional arguments:** An empty text input field with a dropdown arrow on the right.
- JVM Options:** An empty text input field with a dropdown arrow on the right.

At the bottom left, there is a button labeled 'Añadir un nuevo paso' with a dropdown arrow.



Entrega continua (continuous delivery)

- Entrega Continua se entiende como la evolución lógica de la Integración Continua
- La Integración Continua se desarrolla en el ámbito de desarrollo e integración,
- mientras que la Entrega Continua amplía este concepto y lleva la entrega a entornos productivos.



Entrega continua (continuous delivery)

En este negocio, para cuando te has dado cuenta de que tienes problemas, ya es demasiado tarde para salvarte. A no ser que te estés preocupando continuamente, estás acabado.

- Bill Gates