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

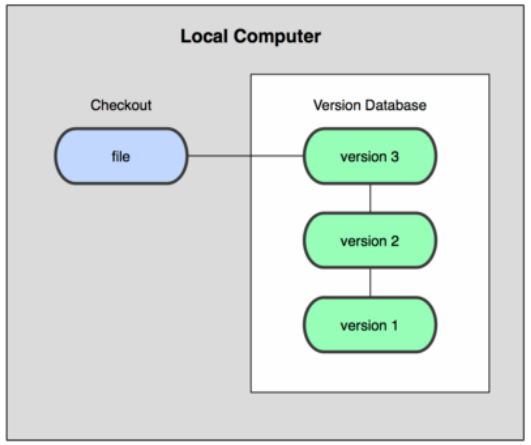
Gestión de código fuente

- ¿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

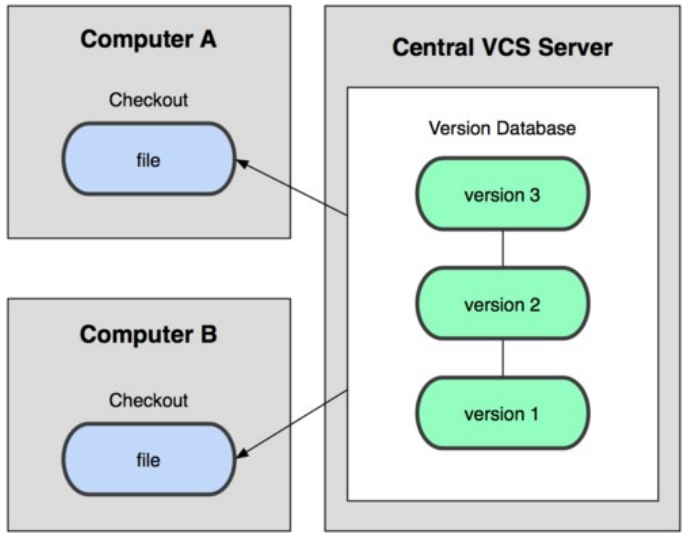
Tipos de sistemas de control de versiones

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

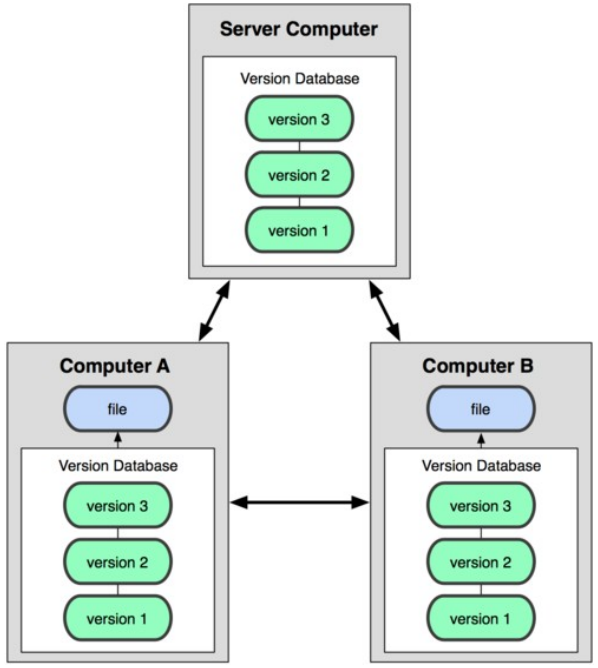
SCV Local (rcs)



SCV centralizado
(cvs, subversion)



SCV distribuido
(git, mercurial)



Gestión de código fuente con Git

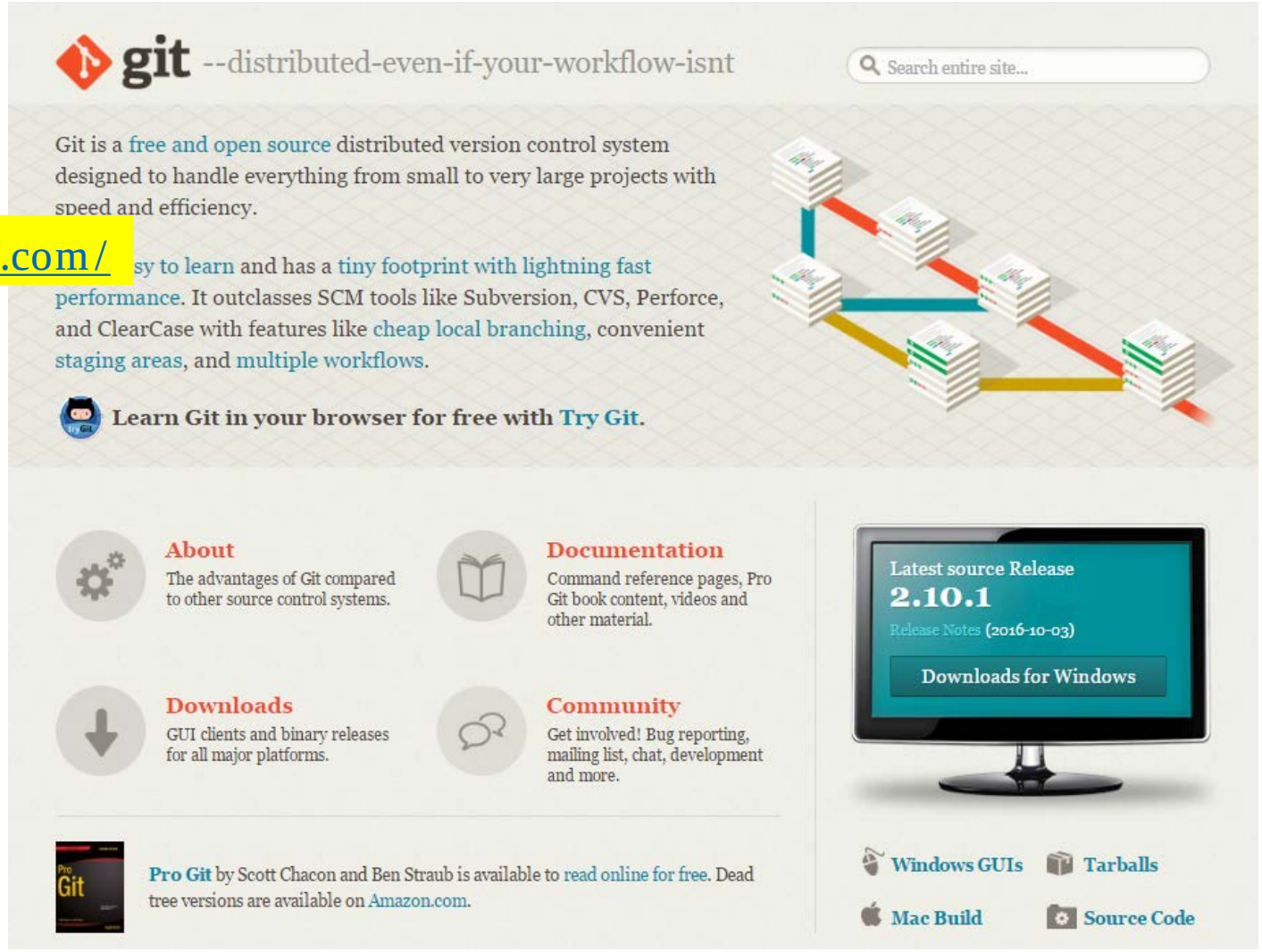
- Ideado por Linus Torvalds para el desarrollo del núcleo de Linux

- 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/)

Sitio web de Git

http://git-scm.com/



Gestión de código fuente con Git

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

- Conceptos básicos de Unix

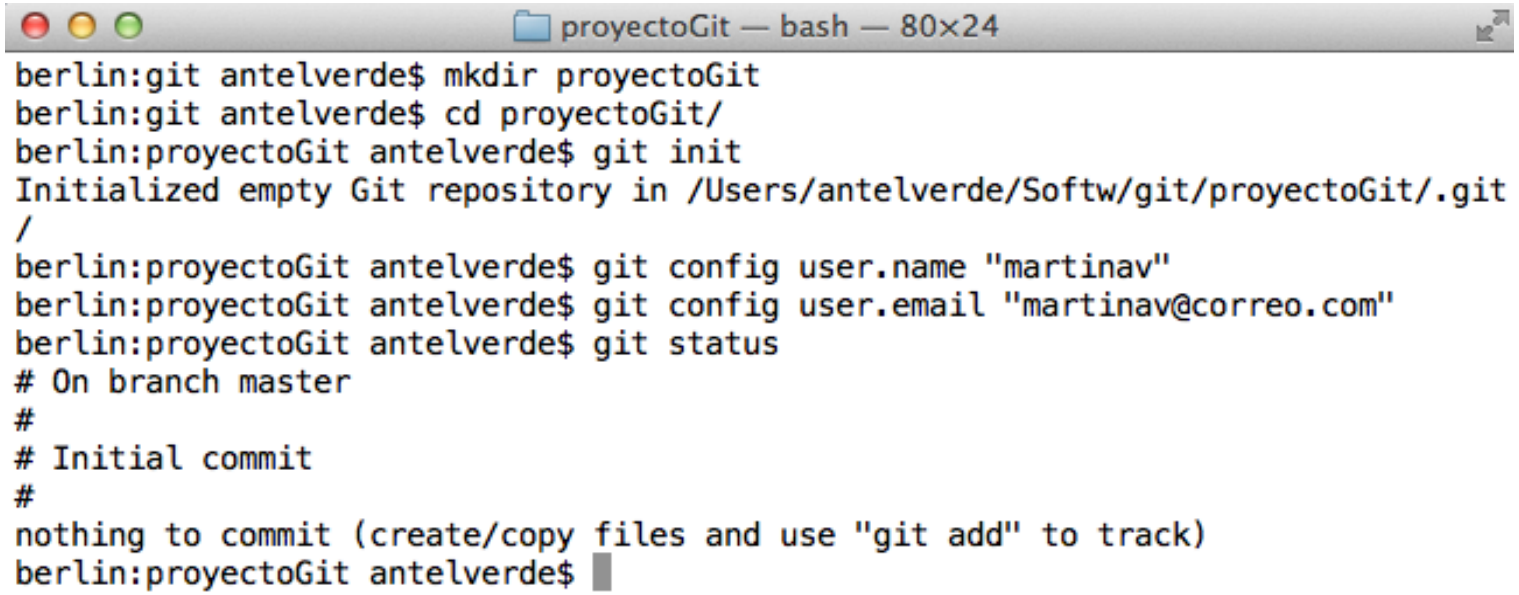
Orden	Operación
`mkdir`	Crear directorio
`cd`	Cambiar directorio
`ls -la`	Listar directorio, modo extendido
`touch`	Crear un fichero
`cat`	Mostrar fichero por pantalla
`git`	Las órdenes para git comienzan por git

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

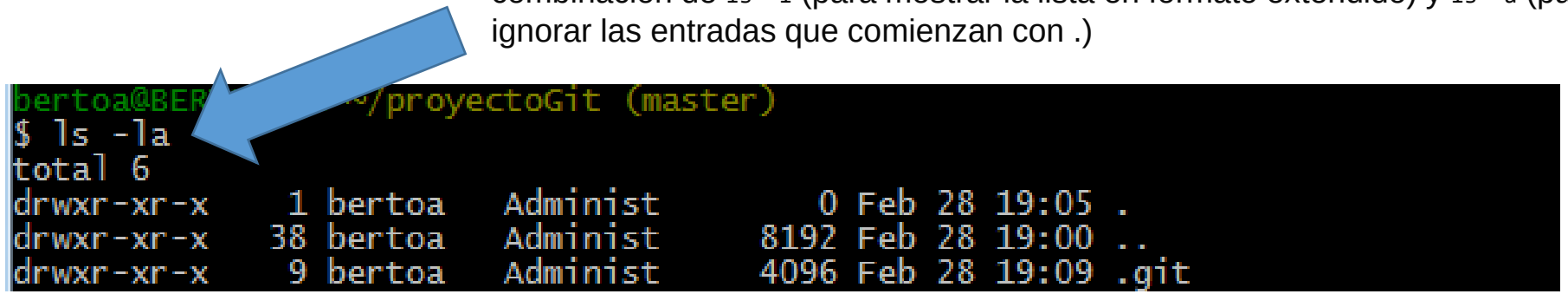
- git init
- git config user.name
- git config user.email



- git config --list

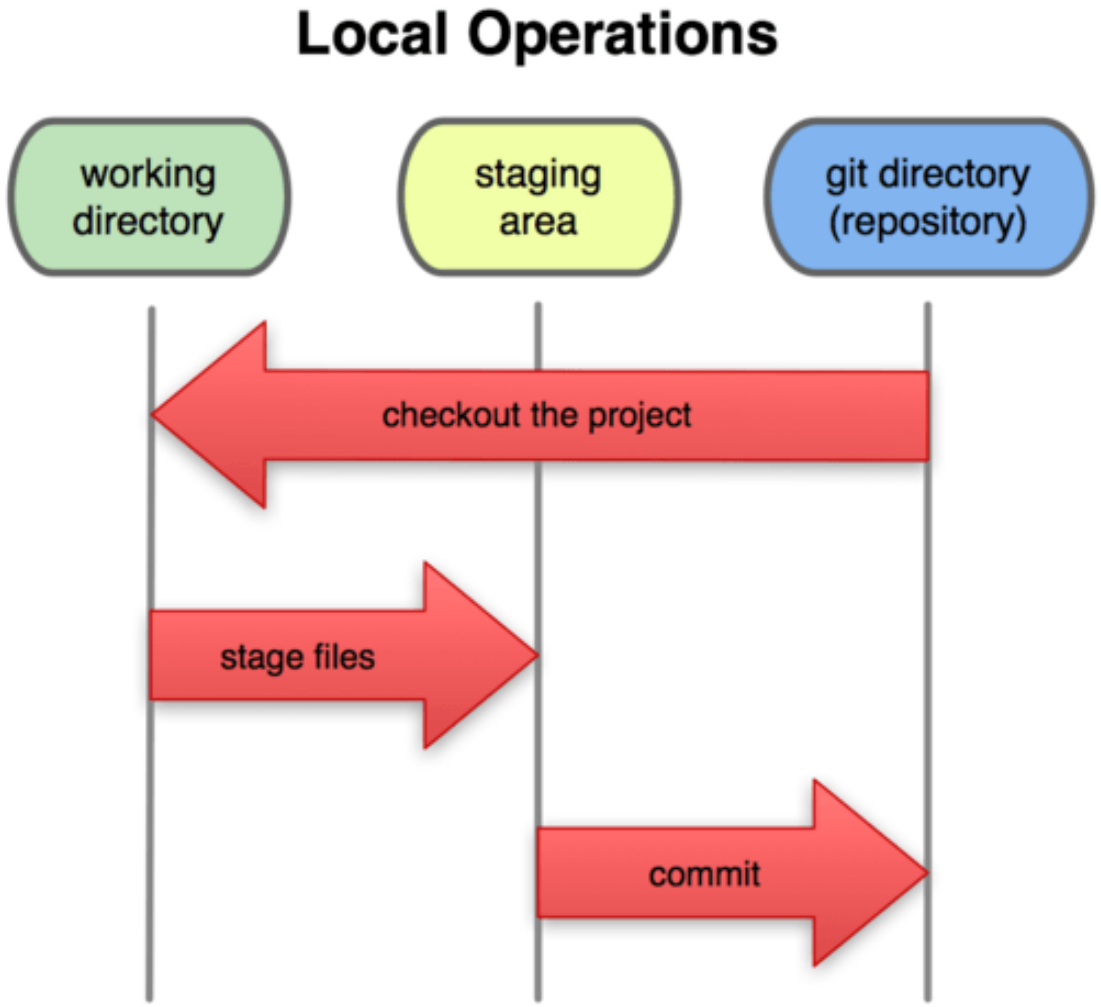
Creación de un repositorio local

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



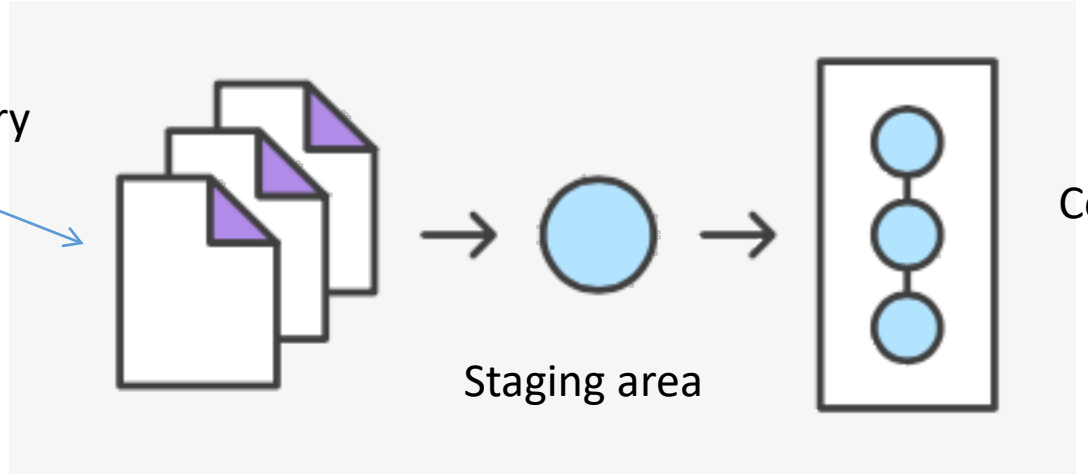
Repositorio git en modo local

- Las tres áreas



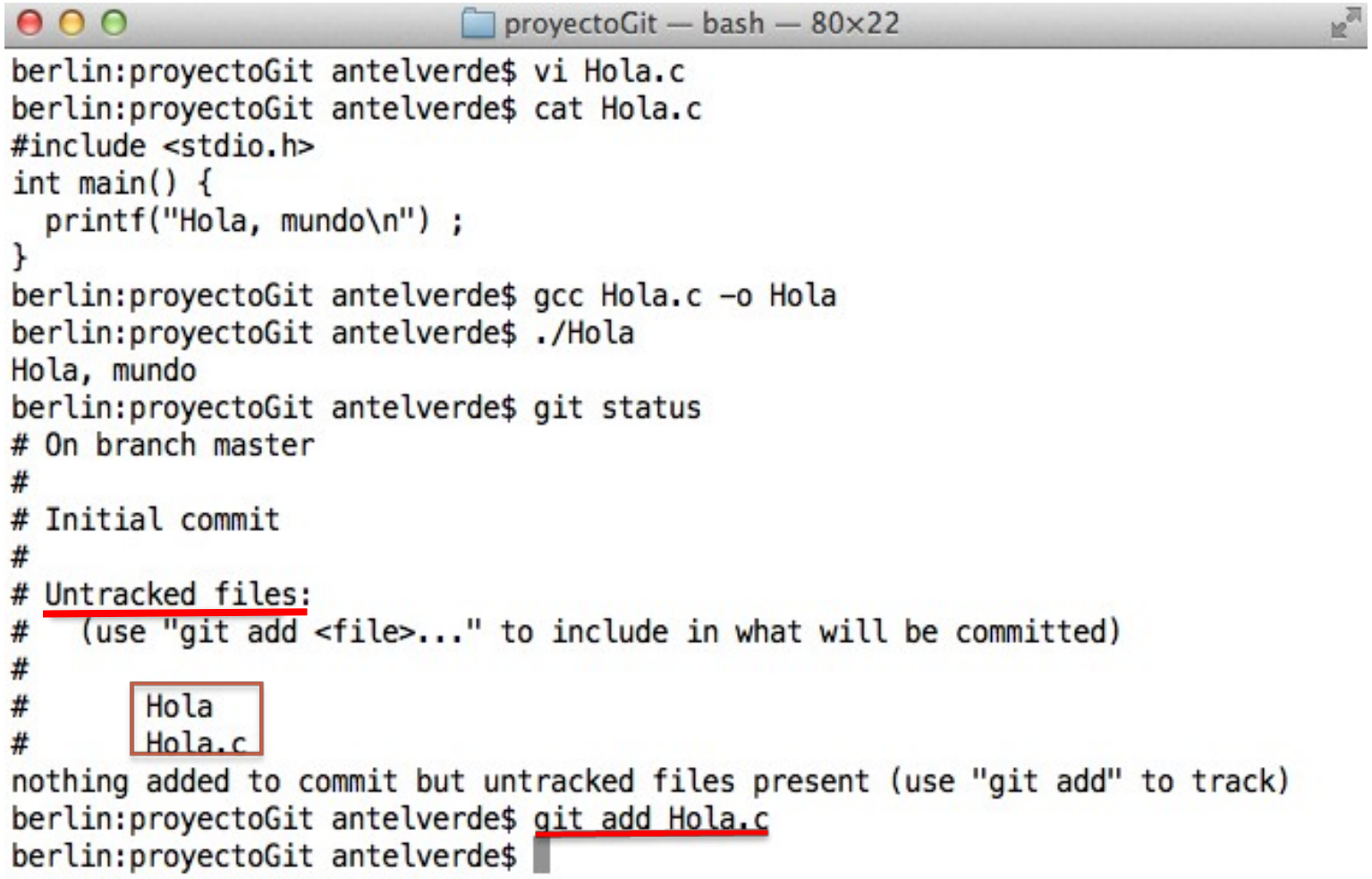
Staging area

- 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

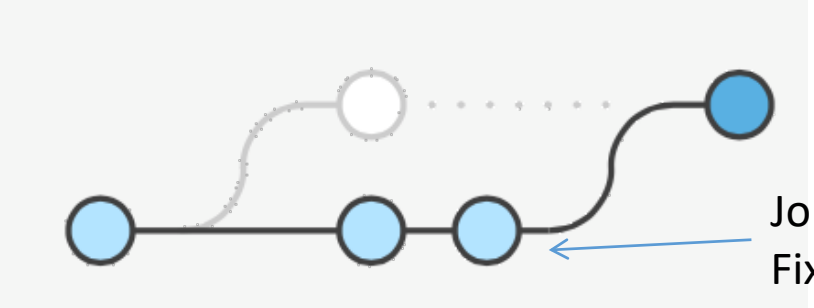
- git add <archivos>



Consolidar cambios en el repositorio

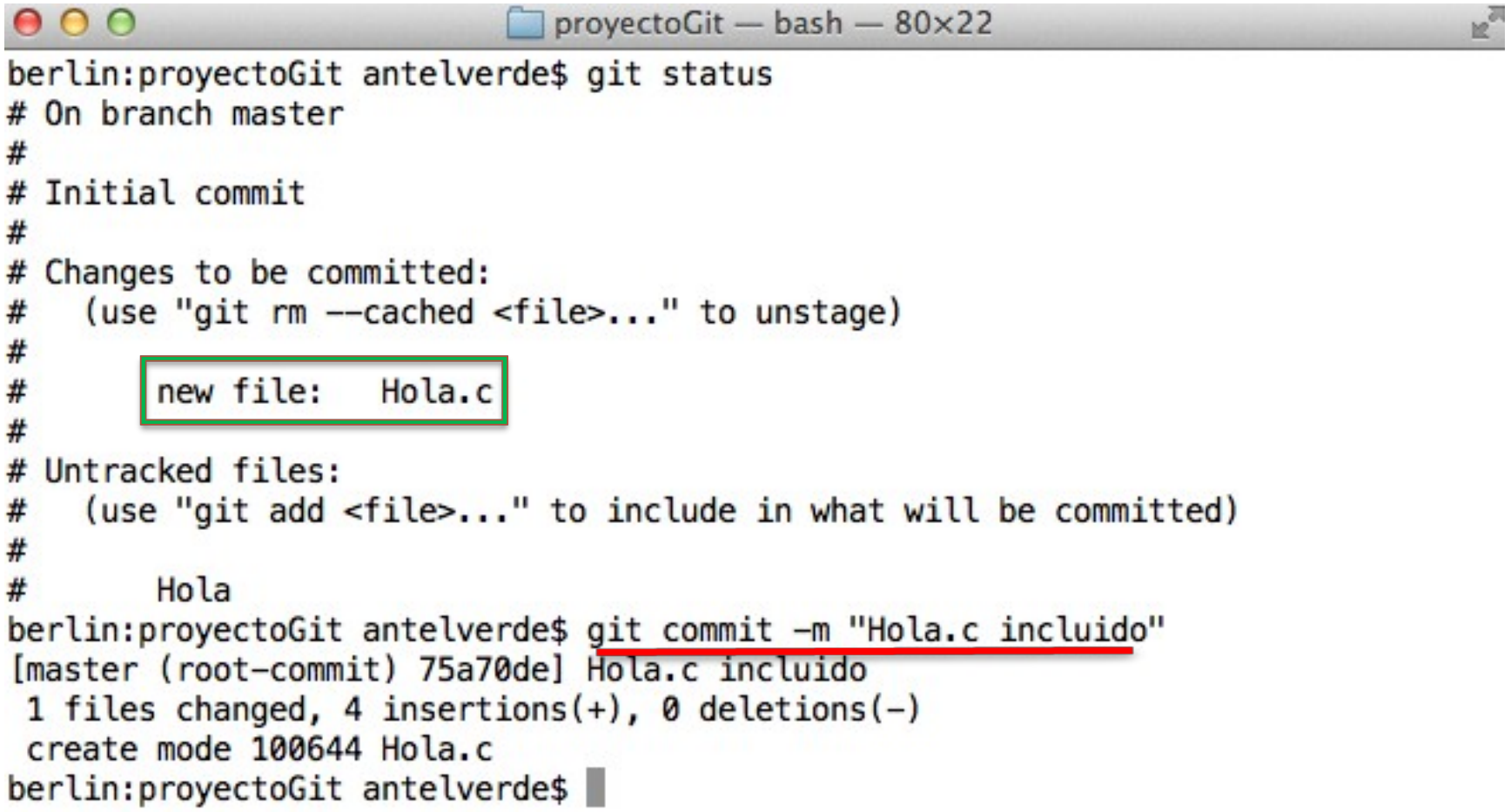
- El commit es uno de los conceptos más importantes en Git

- 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”



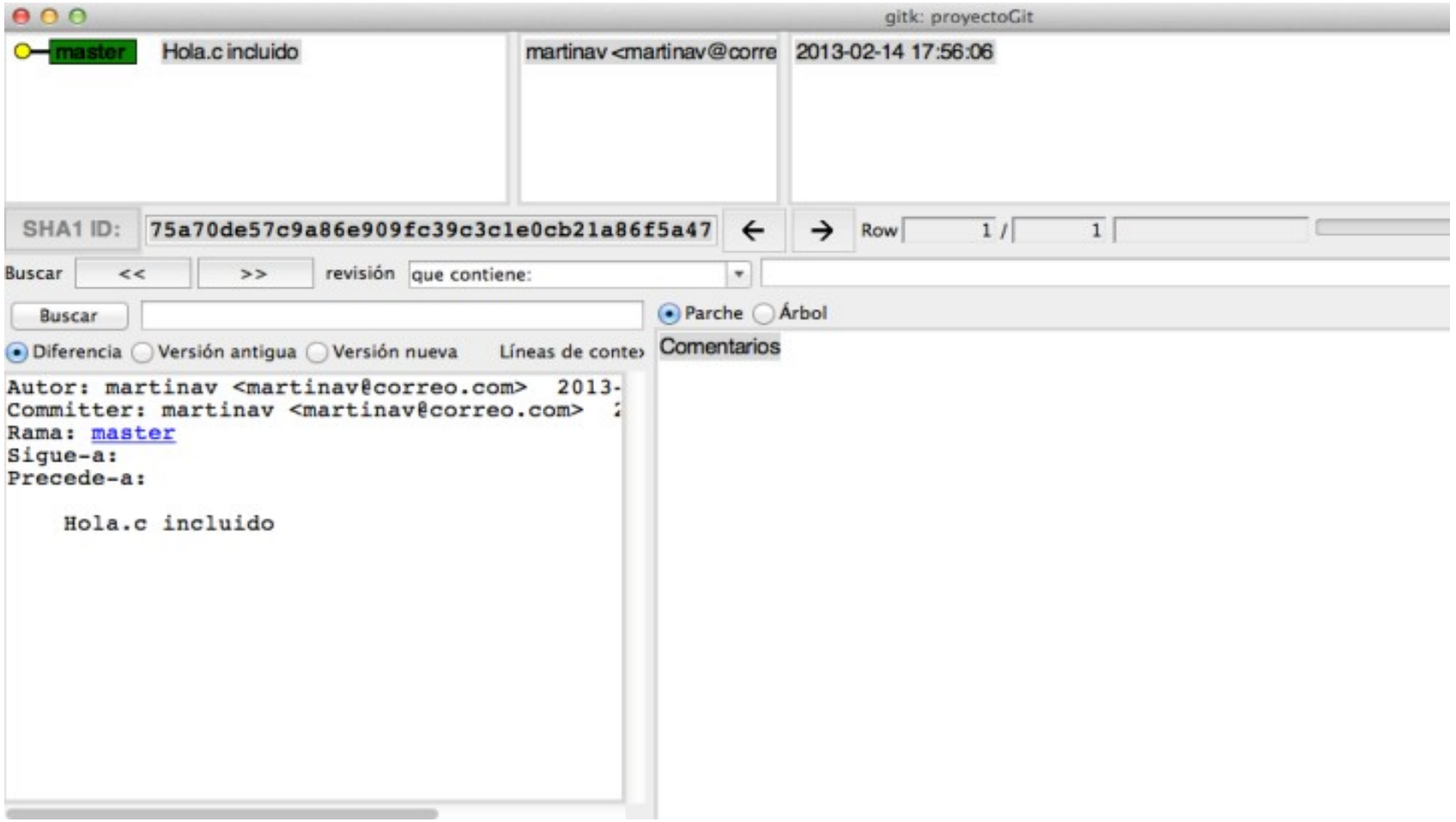
Ejercicio

- 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

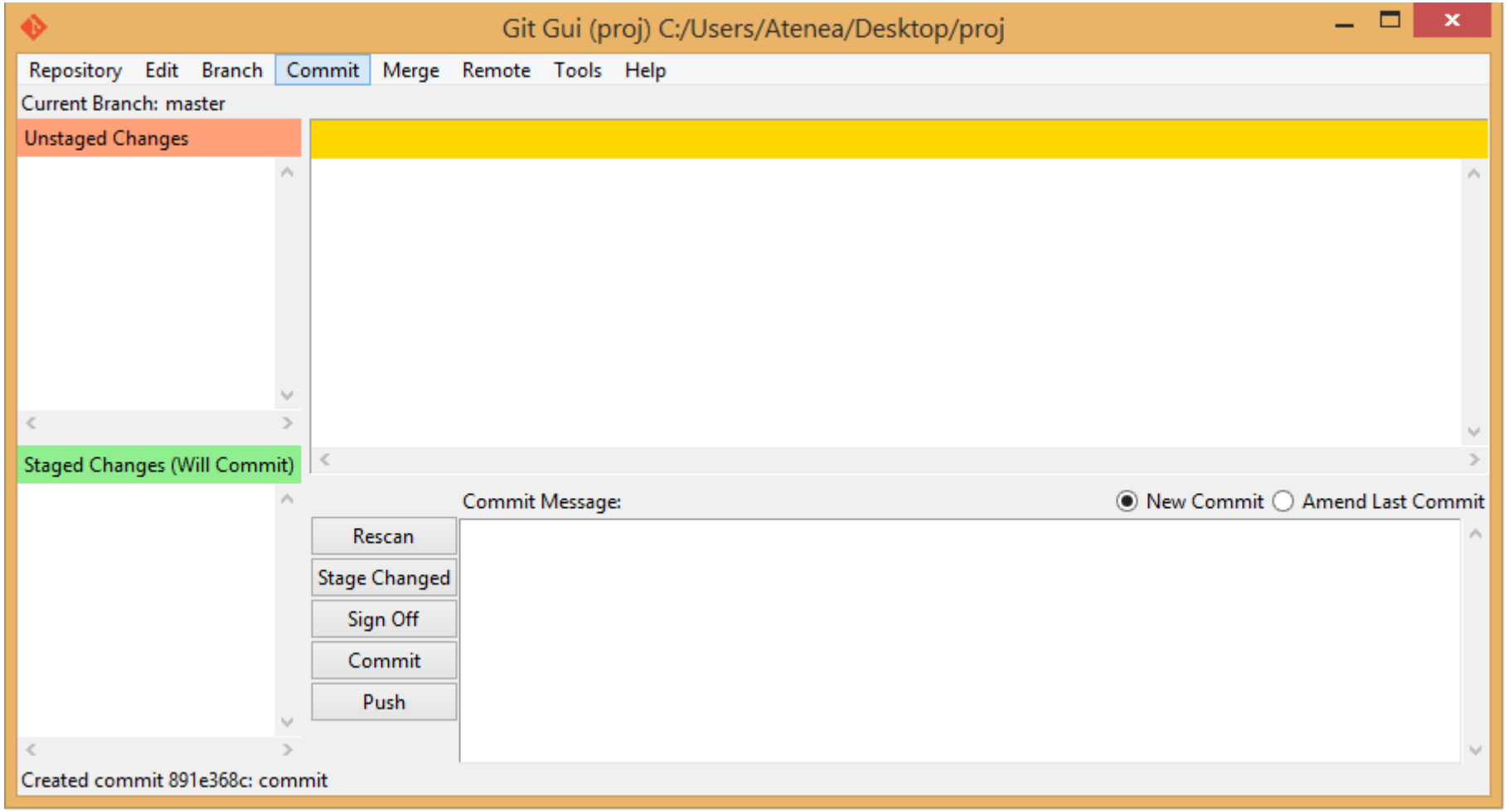
- Buscar información acerca de .gitignore
  - ¿Para qué sirve?
  - ¿Qué formato tiene?

Interfaz gráfica gitk

- gitk --all &



Interfaz gráfica Git GUI



Modificación de un fichero y actualización

```
proyectoGit — bash — 80×24
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

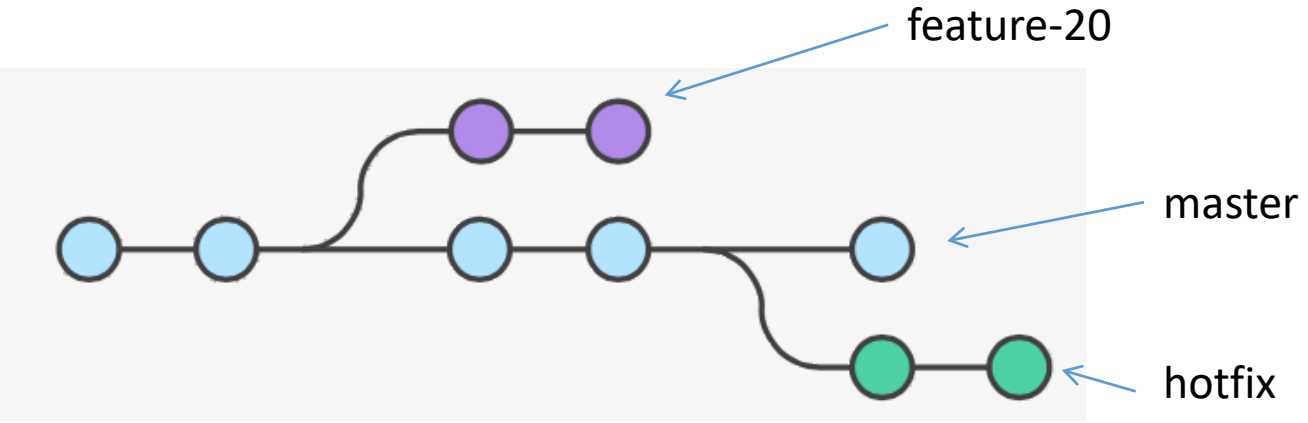
- Añadir un nuevo fichero y ver qué ha ocurrido en Git GUI.
- Añadirlo (add) y volver a observar los cambios en Git GUI.
- Hacer el commit del archivo desde Git GUI.

Viendo los cambios

- git diff <archivo>

Ramas

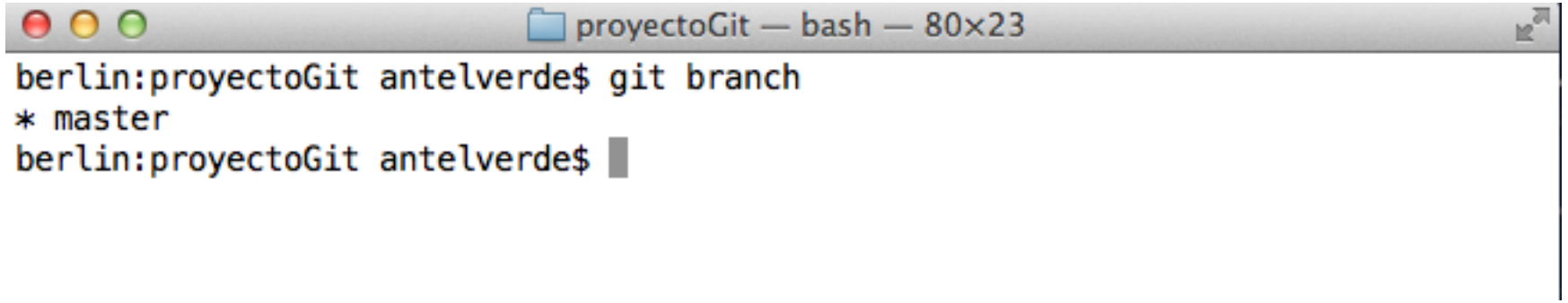
- En Git los commits no tienen porque seguir una secuencia linear sino que puede haber ramas



- A cada rama se le asigna un nombre para distinguirla de la demas
- 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



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

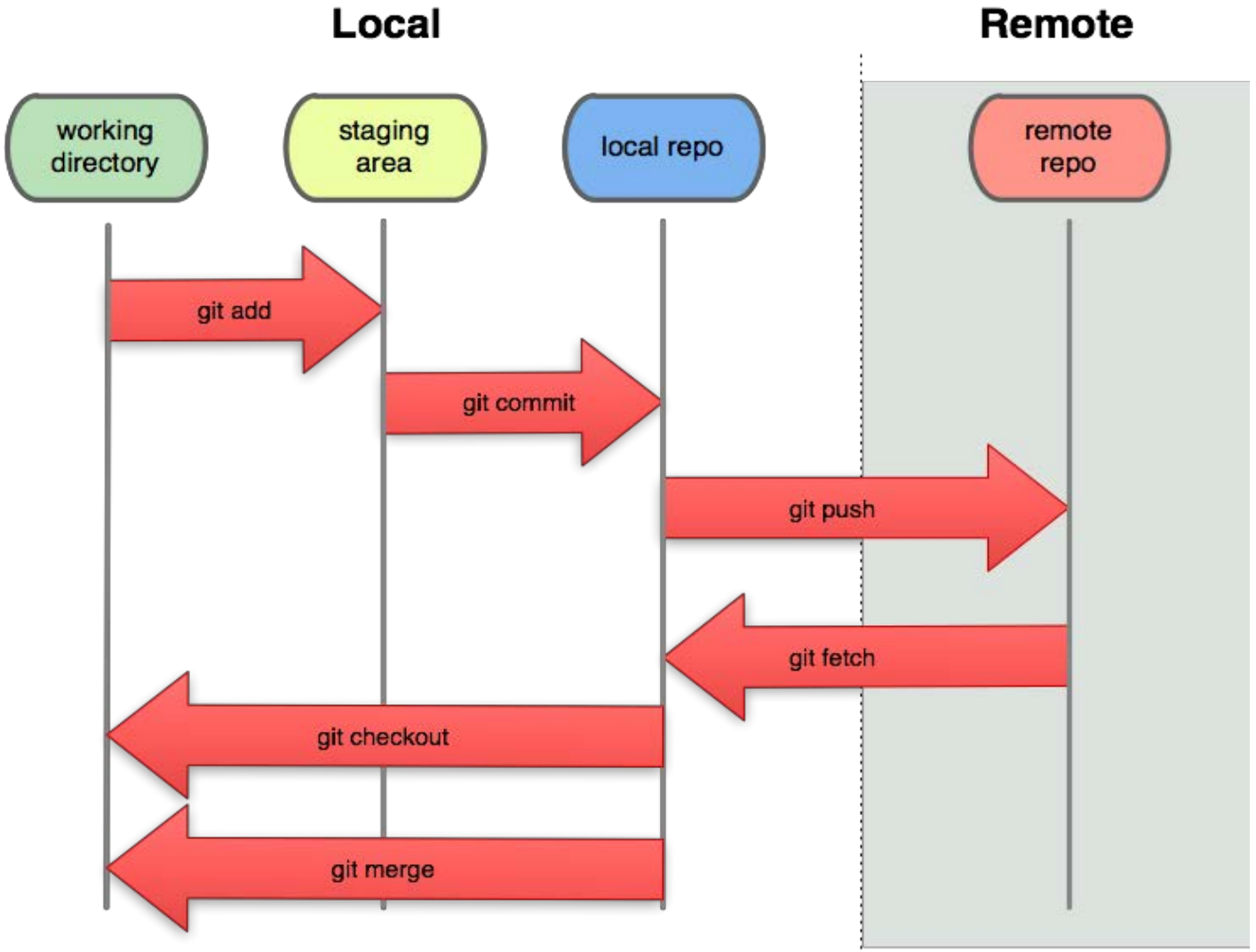
- Apartado 2
  - Creamos un nuevo archivo en la rama testing (lo consolidamos)
  - Creamos un nuevo archivo en la rama master (lo consolidamos)

Ejercicio

- Apartado 3
  - Desde la rama master, fusiona ambas ramas usando:

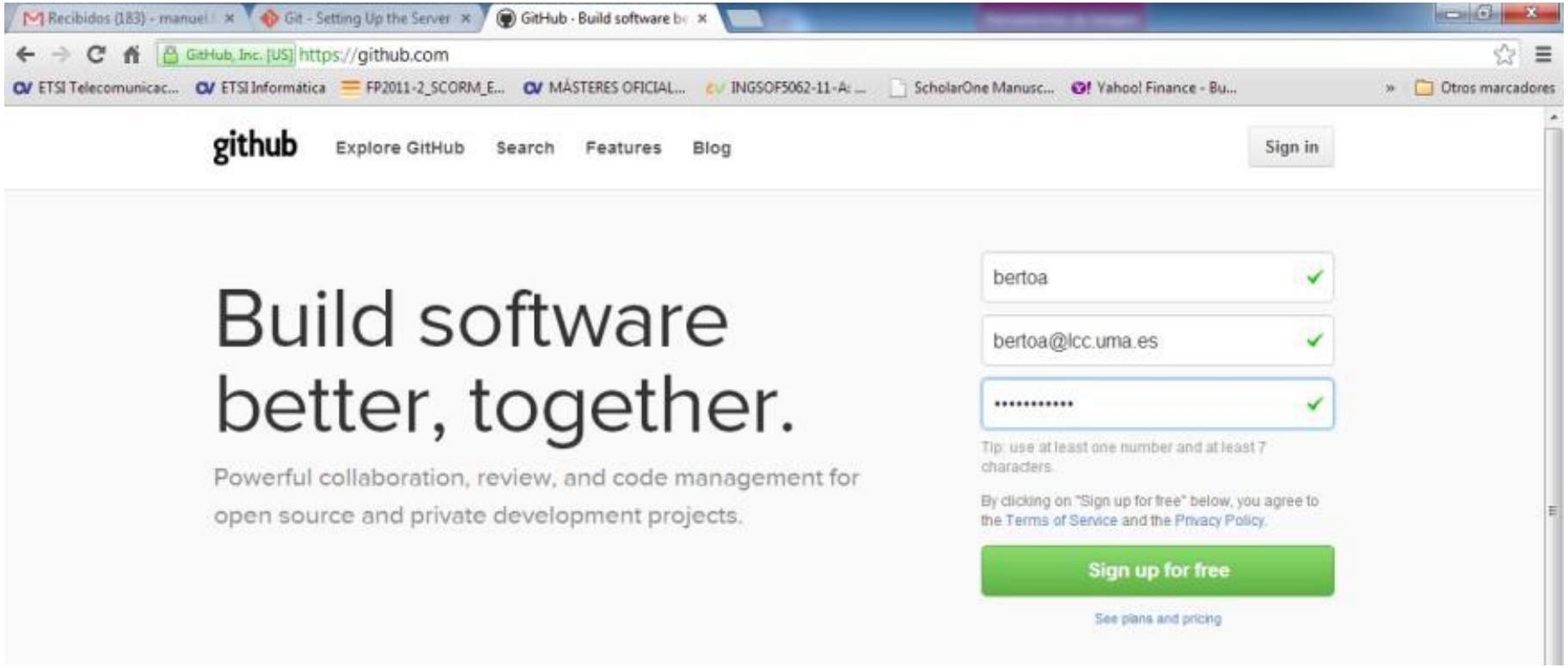
    git merge <nombre rama>
  - Usa gitk para ver todos los cambios realizados
  - Comprueba los archivos para ver qué cambios prevalecen

Git en modo remoto



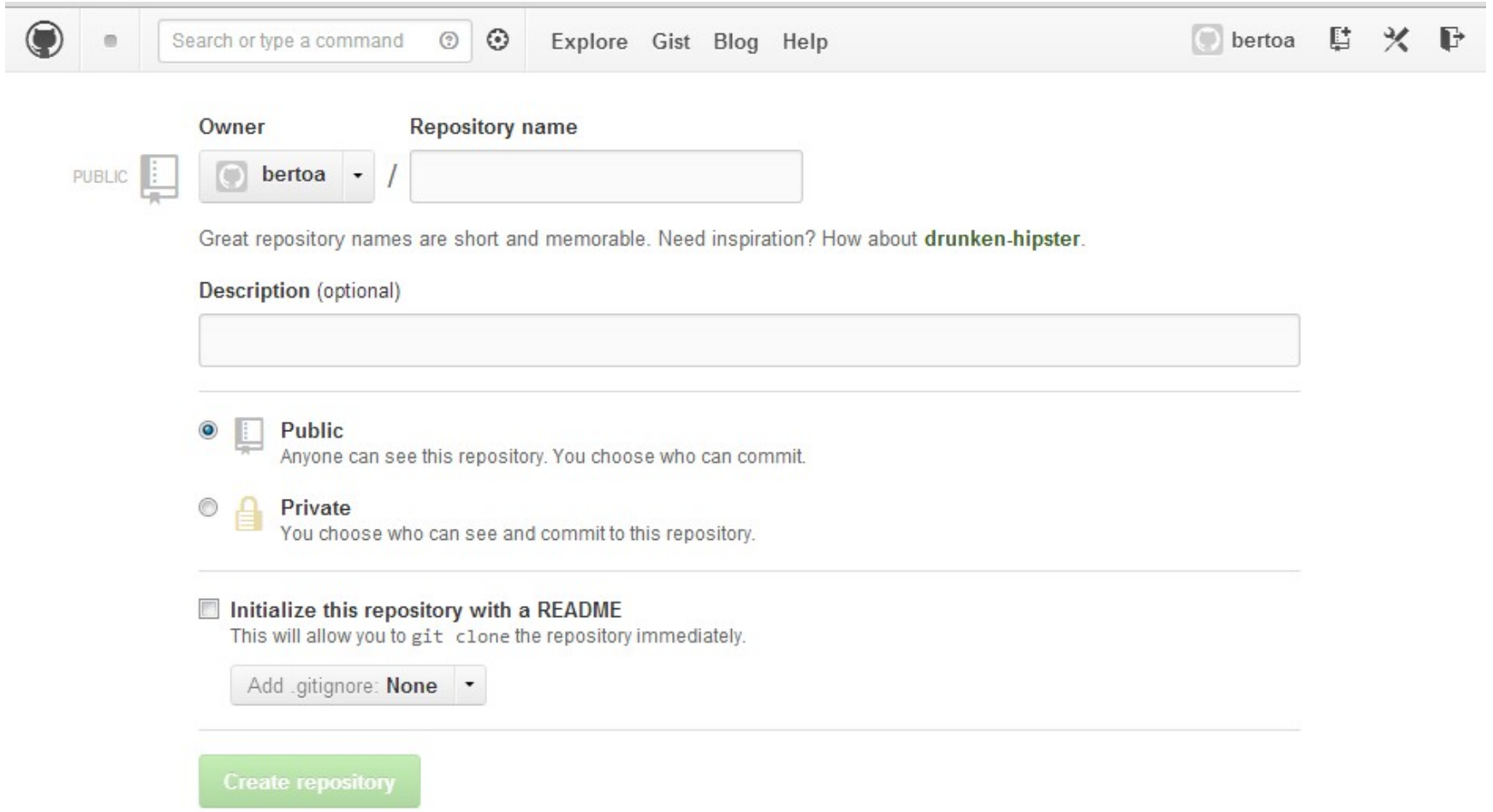
Ejercicio

- Crear una cuenta en GitHub



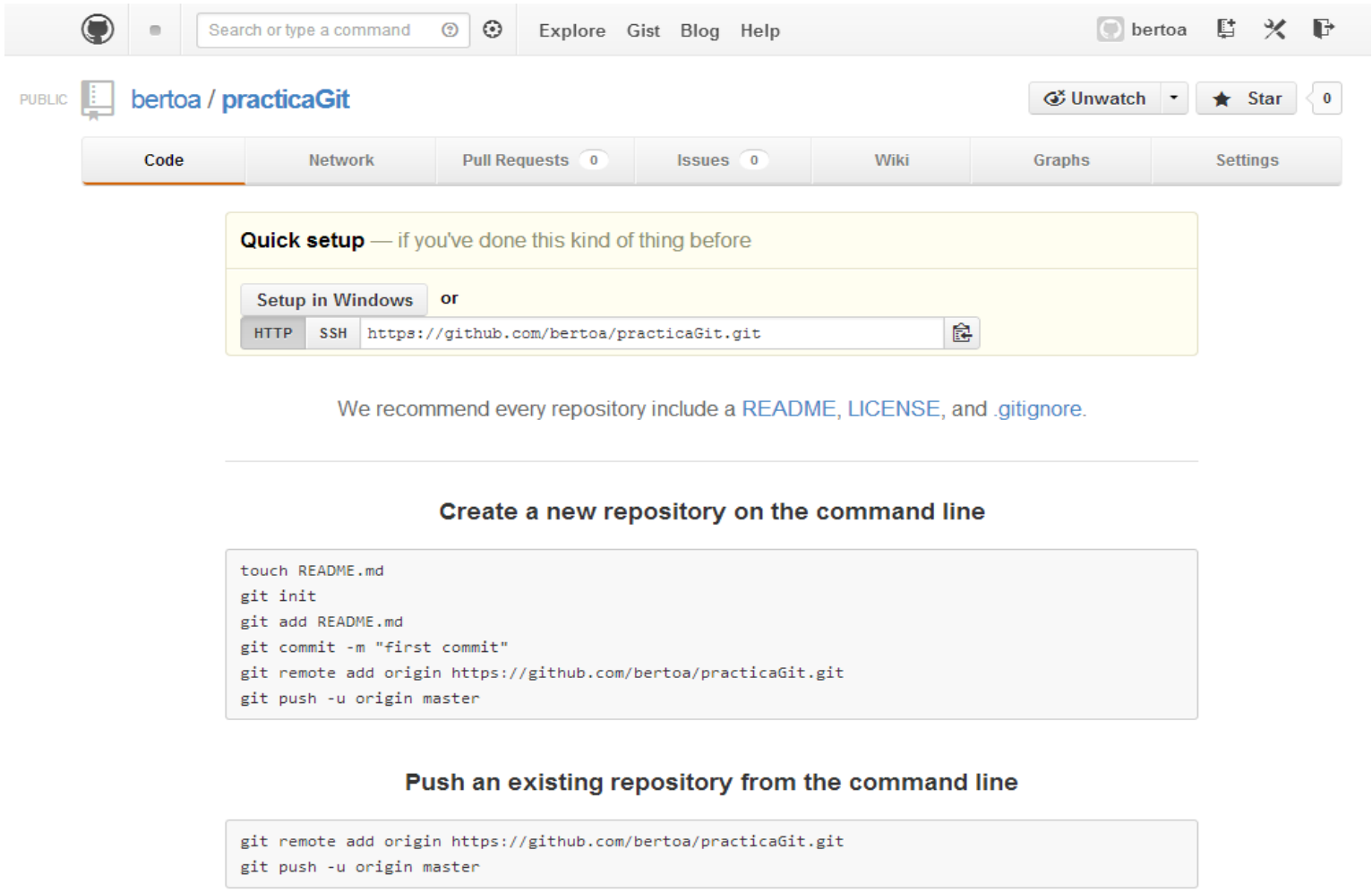
Ejercicio

- Crear un repositorio en GitHub



Ejercicio

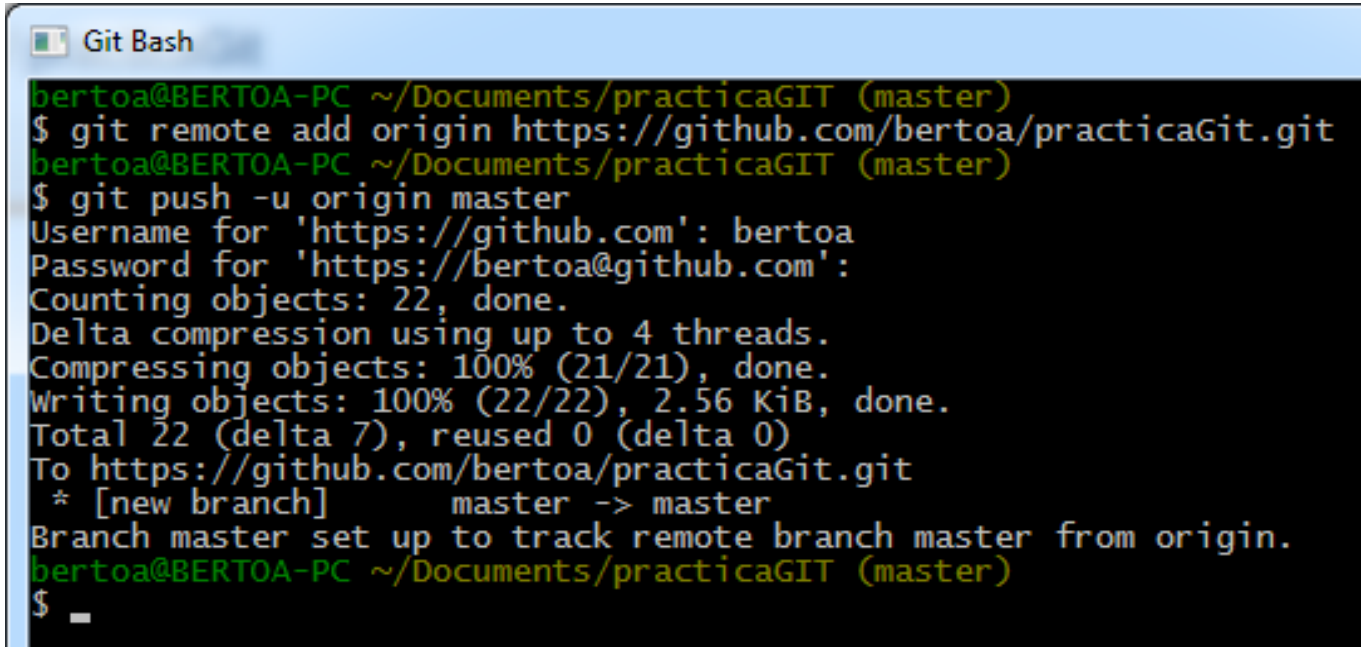
- Subir el repositorio local al remoto



Ejercicio

- Subir el repositorio local al remoto

```
git remote add origin https://github.com/bertoa/practicaGit.git
git push -u origin 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 me
    - 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("Aqui 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>
```

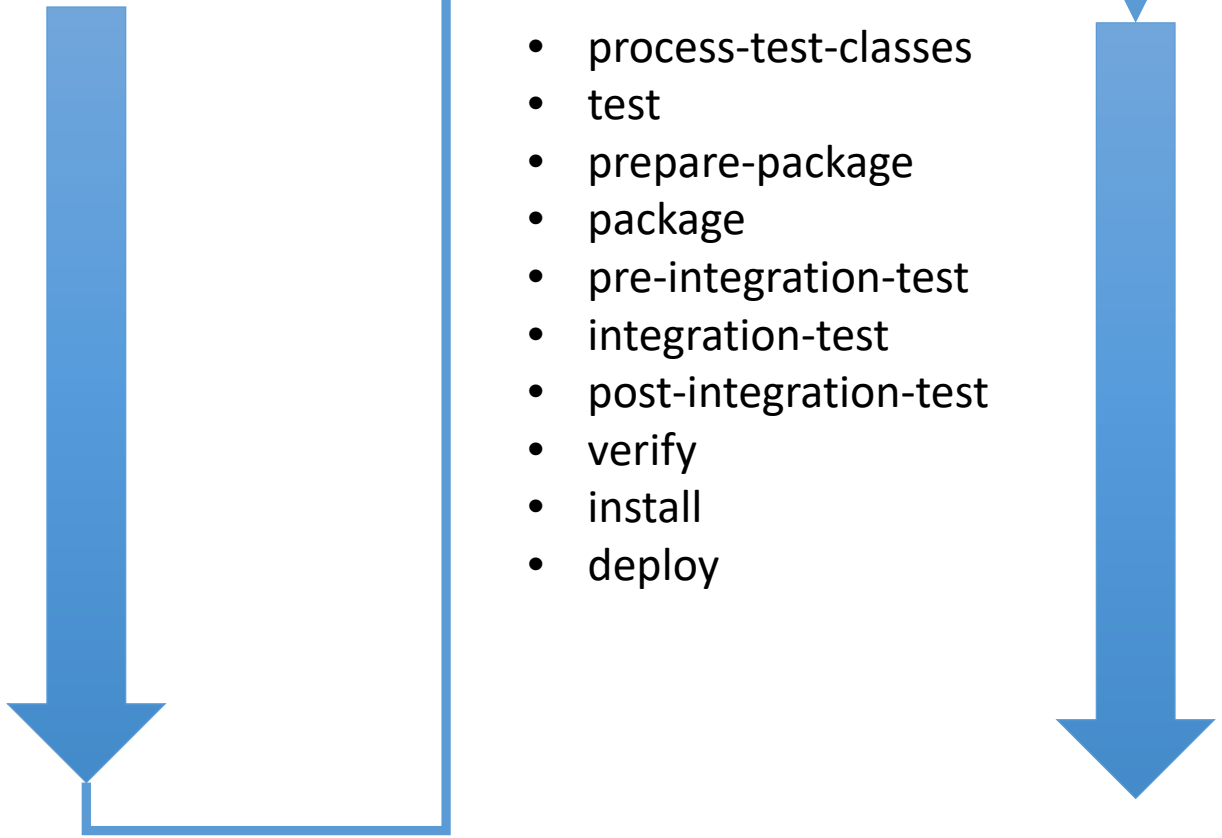
- Cuando se especifica una dependencia en el POM, Maven la descarga automáticamente y la resuelve
- Todas las dependencias se guardan en un repositorio local por lo que solo las descarga una vez

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

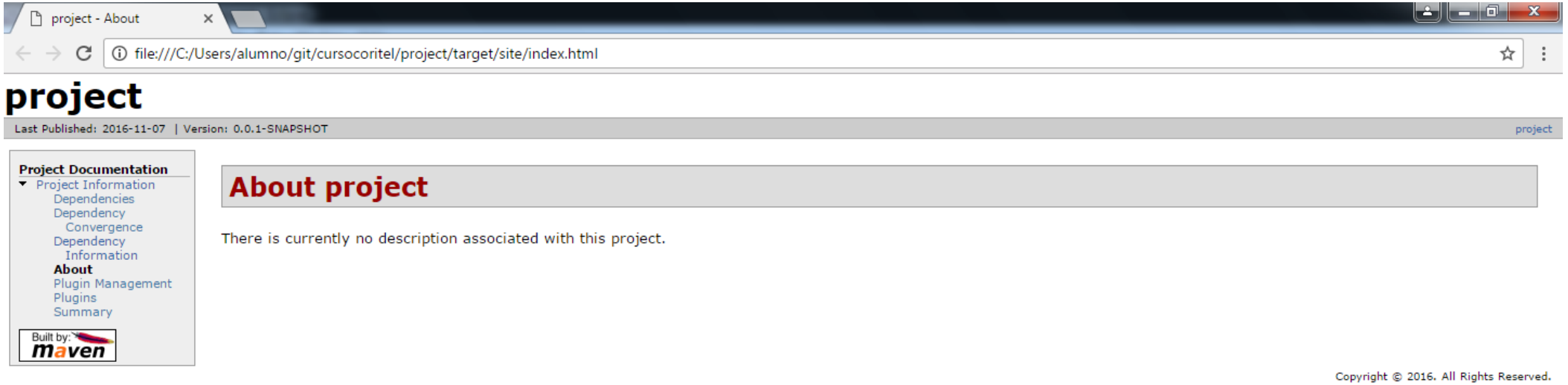
Cuando se le proporciona una fase a Maven, Maven ejecuta todas las fases desde la primera hasta que alcanza la misma

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

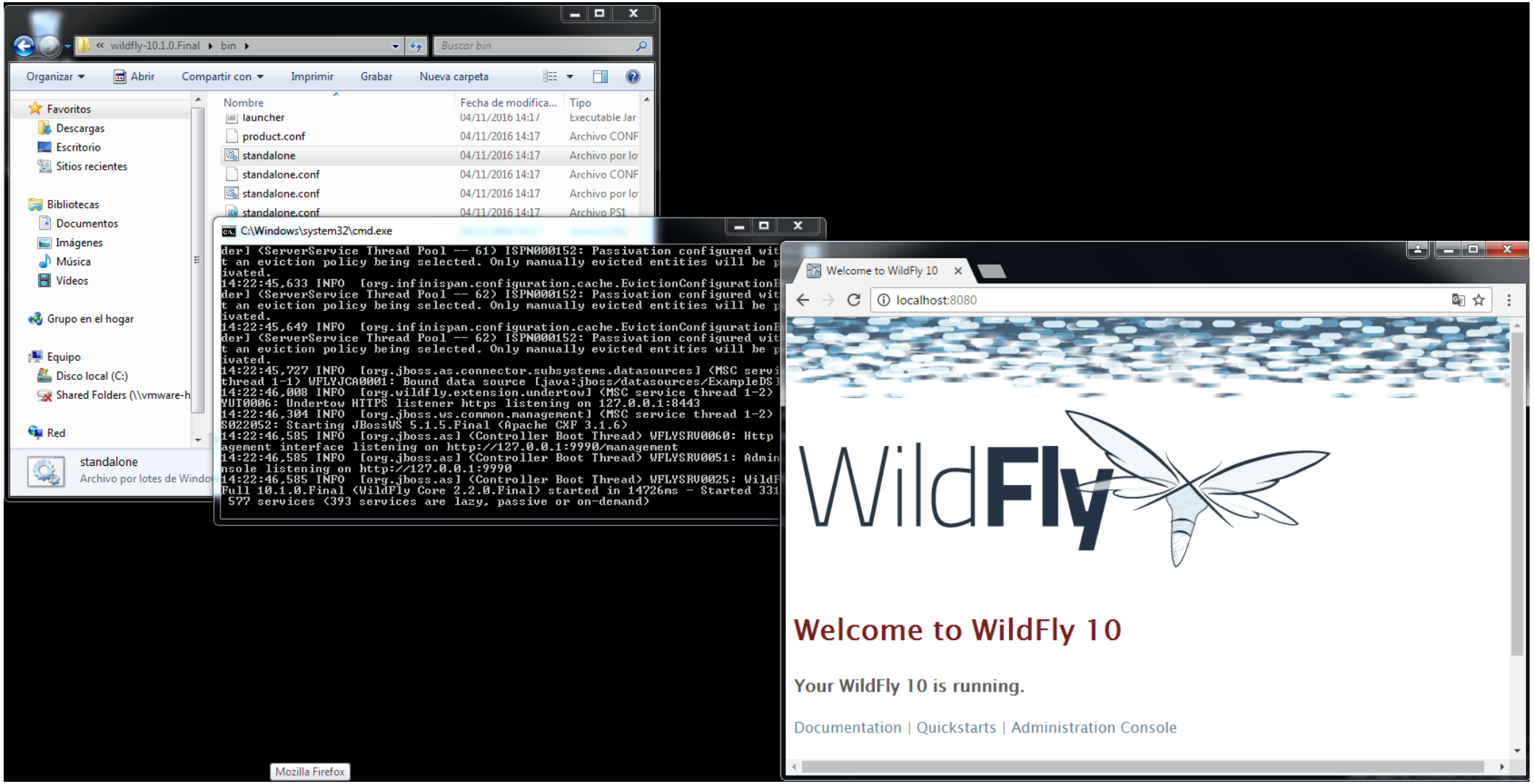
Jenkins

- 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

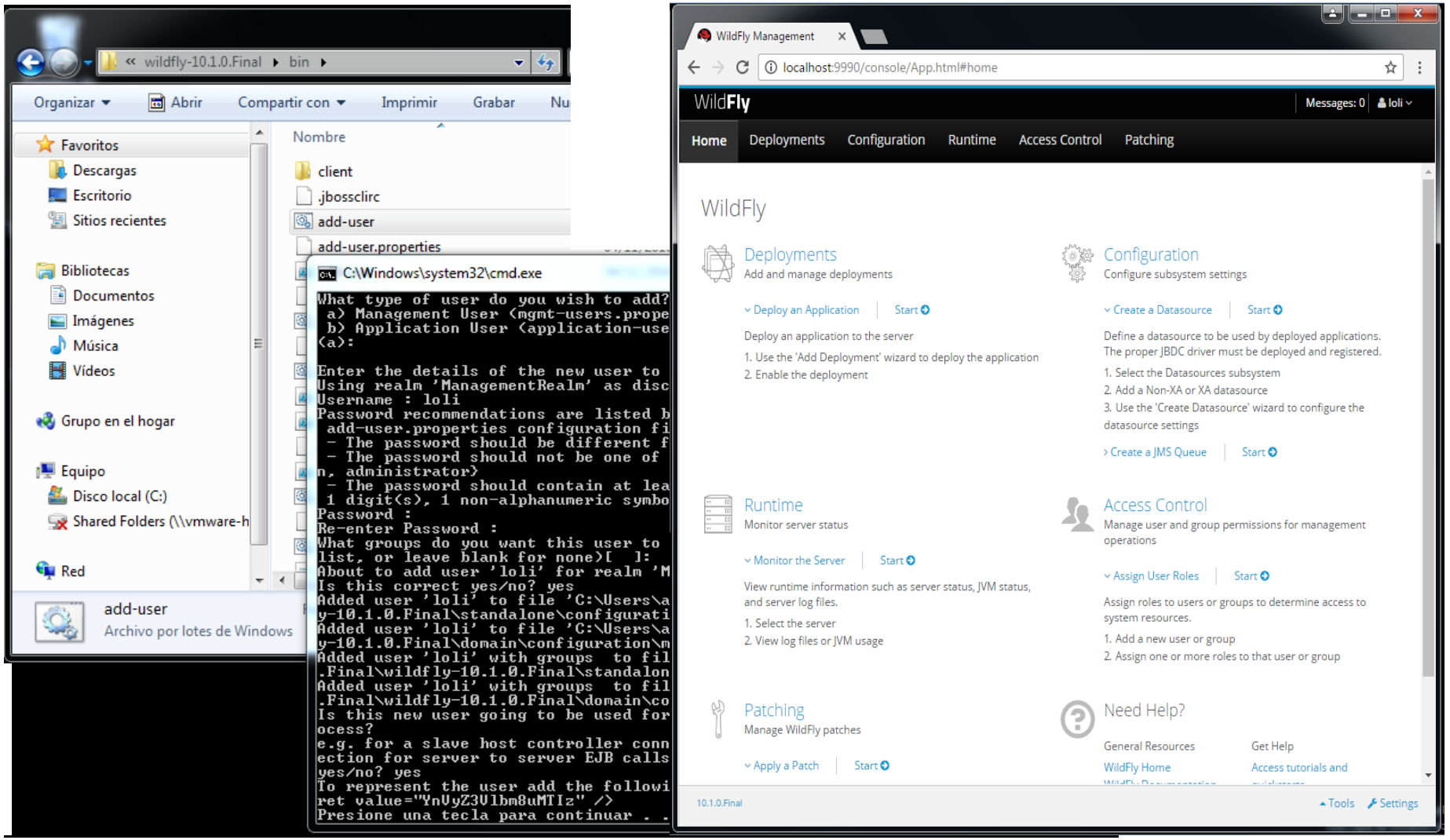
En nuestro caso…

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

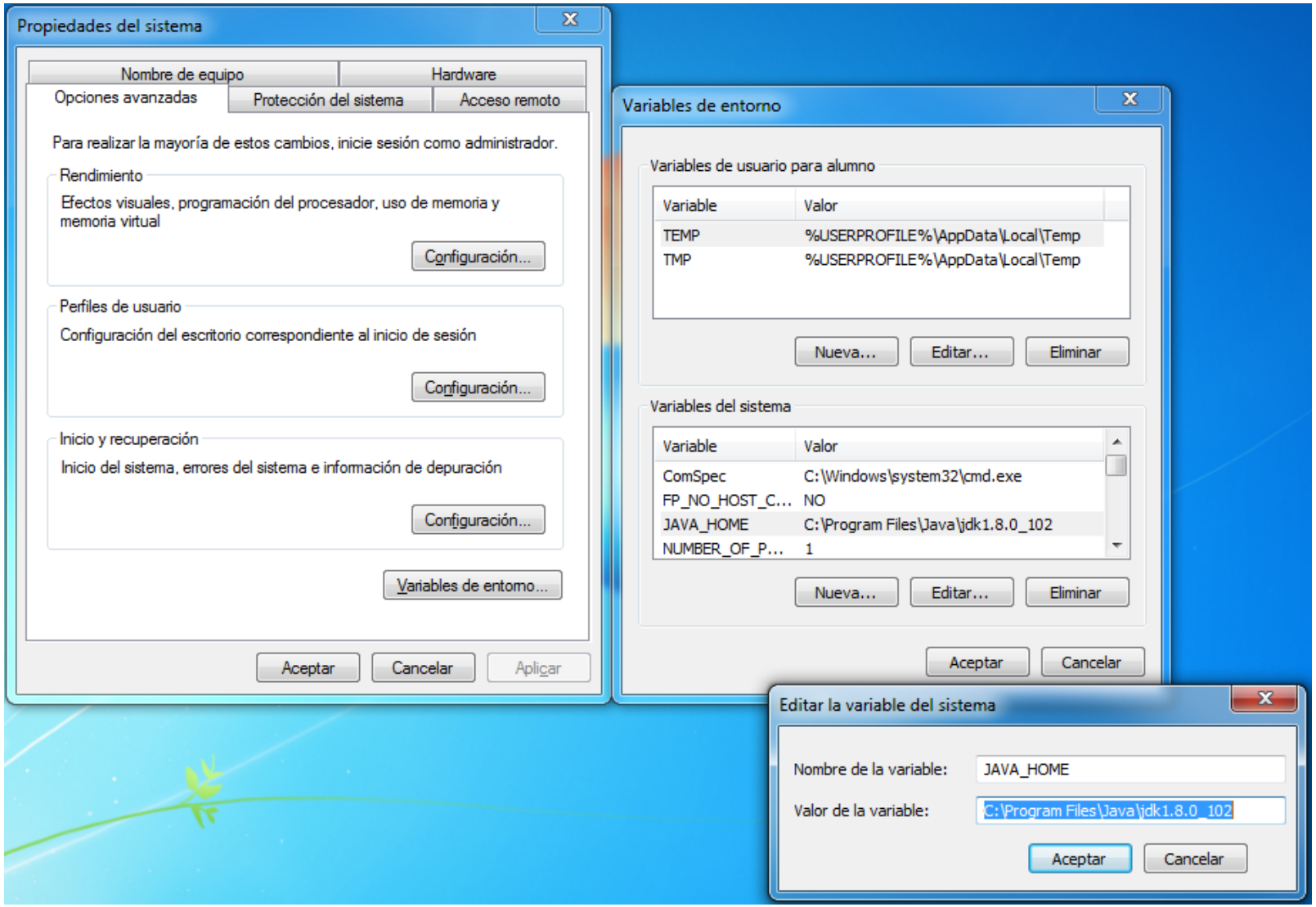

wildfly-10.1.0.Final ▸ bin ▸
Buscar bin
Organizar
Abrir
Compartir con
Imprimir
Grabar
Nueva carpeta
Favoritos
Descargas
Escritorio
Sitios recientes
Bibliotecas
Documentos
Imágenes
Música
Vídeos
Grupo en el hogar
Equipo
Disco local (C:)
Shared Folders (\\vmware-h
Red
Nombre
Fecha de modifica...
Tipo
launcher
product.conf
standalone
standalone.conf
standalone.conf
standalone.conf
Archivo CONF
Archivo por lo
Archivo PS1
standalone
Archivo por lotes de Windo
C:\Windows\system32\cmd.exe
14:22:45,633 INFO [org.infinispan.configuration.cache.EvictionConfiguration
14:22:45,649 INFO [org.infinispan.configuration.cache.EvictionConfiguration
14:22:45,727 INFO [org.jboss.as.connector.subsystems.datasources] (MSC servi
14:22:46,008 INFO [org.wildfly.extension.undertow] (MSC service thread 1-2)
14:22:46,304 INFO [org.jboss.ws.common.management] (MSC service thread 1-2)
14:22:46,585 INFO [org.jboss.as] (Controller Boot Thread) WFLYSRV0060: Http
14:22:46,585 INFO [org.jboss.as] (Controller Boot Thread) WFLYSRV0051: Admin
14:22:46,585 INFO [org.jboss.as] (Controller Boot Thread) WFLYSRV0025: WildF
577 services (393 services are lazy, passive or on-demand)
Welcome to WildFly 10
localhost:8080
WildFly
Welcome to WildFly 10
Your WildFly 10 is running.
Documentation | Quickstarts | Administration Console
Mozilla Firefox

Jenkins


wildfly-10.1.0.Final ▸ bin ▸
Organizar
Abrir
Compartir con
Imprimir
Grabar
Favoritos
Descargas
Escritorio
Sitios recientes
Bibliotecas
Documentos
Imágenes
Música
Vídeos
Grupo en el hogar
Equipo
Disco local (C:)
Shared Folders (\\vmware-h
Red
Nombre
client
.jbossclirc
add-user
add-user.properties
add-user
Archivo por lotes de Windows
C:\Windows\system32\cmd.exe
What type of user do you wish to add?
a) Management User (mgmt-users.prope
b) Application User (application-use
(a):
Enter the details of the new user to
Using realm 'ManagementRealm' as disc
Username : loli
Password recommendations are listed b
add-user.properties configuration fi
- The password should be different f
- The password should not be one of
n, administrator)
- The password should contain at lea
1 digit(s), 1 non-alphanumeric symbo
Password :
Re-enter Password :
What groups do you want this user to
list, or leave blank for none)[ ]:
About to add user 'loli' for realm 'M
Is this correct yes/no? yes
Added user 'loli' to file 'C:\Users\a
y-10.1.0.Final\standalone\configurati
Added user 'loli' to file 'C:\Users\a
y-10.1.0.Final\domain\configuration\m
Added user 'loli' with groups to fil
.Final\wildfly-10.1.0.Final\standalon
Added user 'loli' with groups to fil
.Final\wildfly-10.1.0.Final\domain\co
Is this new user going to be used for
ocess?
e.g. for a slave host controller conn
ection for server to server EJB calls
yes/no? yes
To represent the user add the followi
ret value="YnVyZ3Vlbm8uMTIz" />
Presione una tecla para continuar . .
WildFly Management
localhost:9990/console/App.html#home
WildFly
Messages: 0
loli
Home
Deployments
Configuration
Runtime
Access Control
Patching
WildFly
Deployments
Add and manage deployments
Deploy an Application
Start
Deploy an application to the server
1. Use the 'Add Deployment' wizard to deploy the application
2. Enable the deployment
Configuration
Configure subsystem settings
Create a Datasource
Start
Define a datasource to be used by deployed applications.
The proper JBDC driver must be deployed and registered.
1. Select the Datasources subsystem
2. Add a Non-XA or XA datasource
3. Use the 'Create Datasource' wizard to configure the datasource settings
Create a JMS Queue
Start
Runtime
Monitor server status
Monitor the Server
Start
View runtime information such as server status, JVM status, and server log files.
1. Select the server
2. View log files or JVM usage
Access Control
Manage user and group permissions for management operations
Assign User Roles
Start
Assign roles to users or groups to determine access to system resources.
1. Add a new user or group
2. Assign one or more roles to that user or group
Patching
Manage WildFly patches
Apply a Patch
Start
Need Help?
General Resources
Get Help
WildFly Home
Access tutorials and
10.1.0.Final
Tools
Settings

Jenkins

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

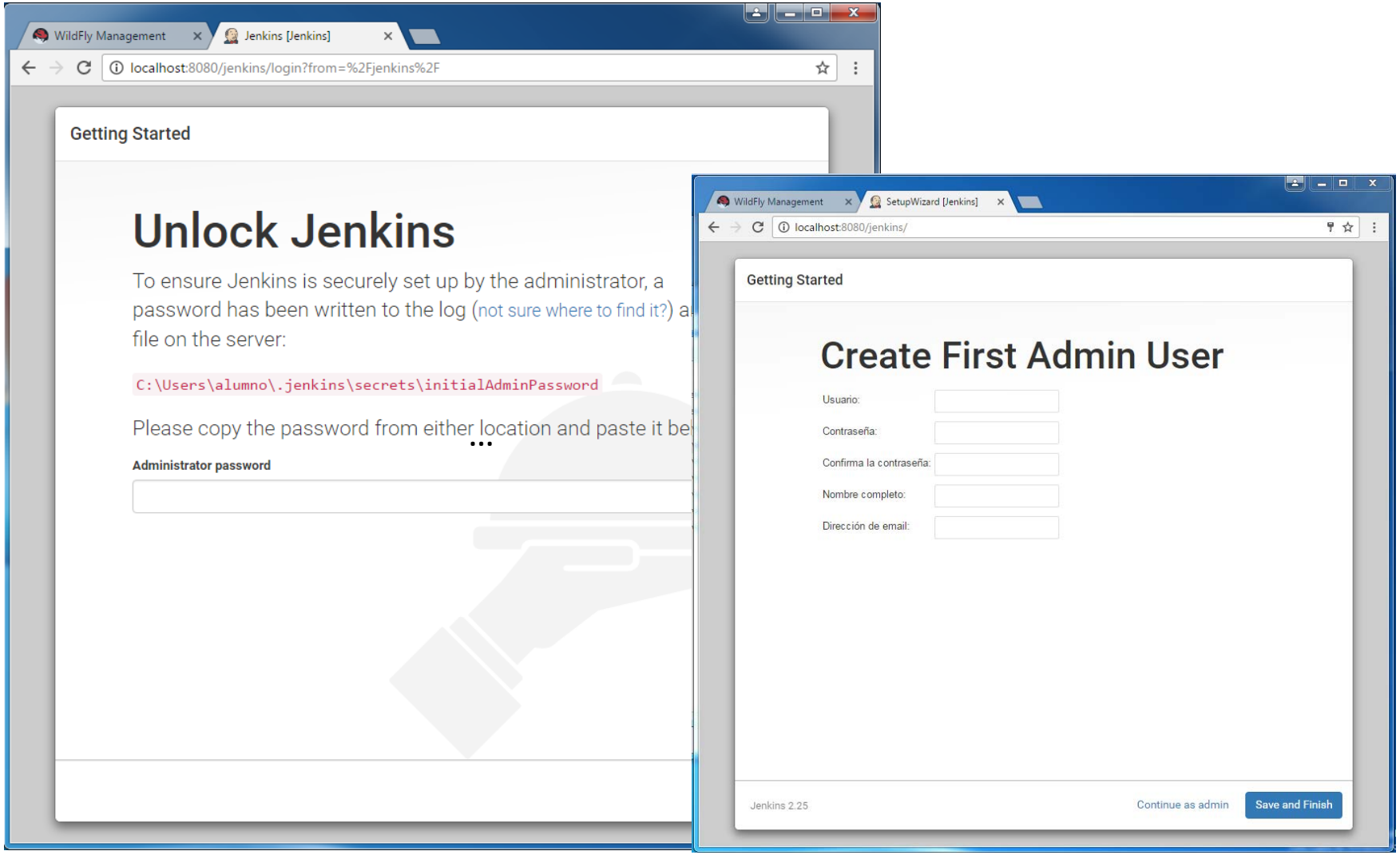


Jenkins

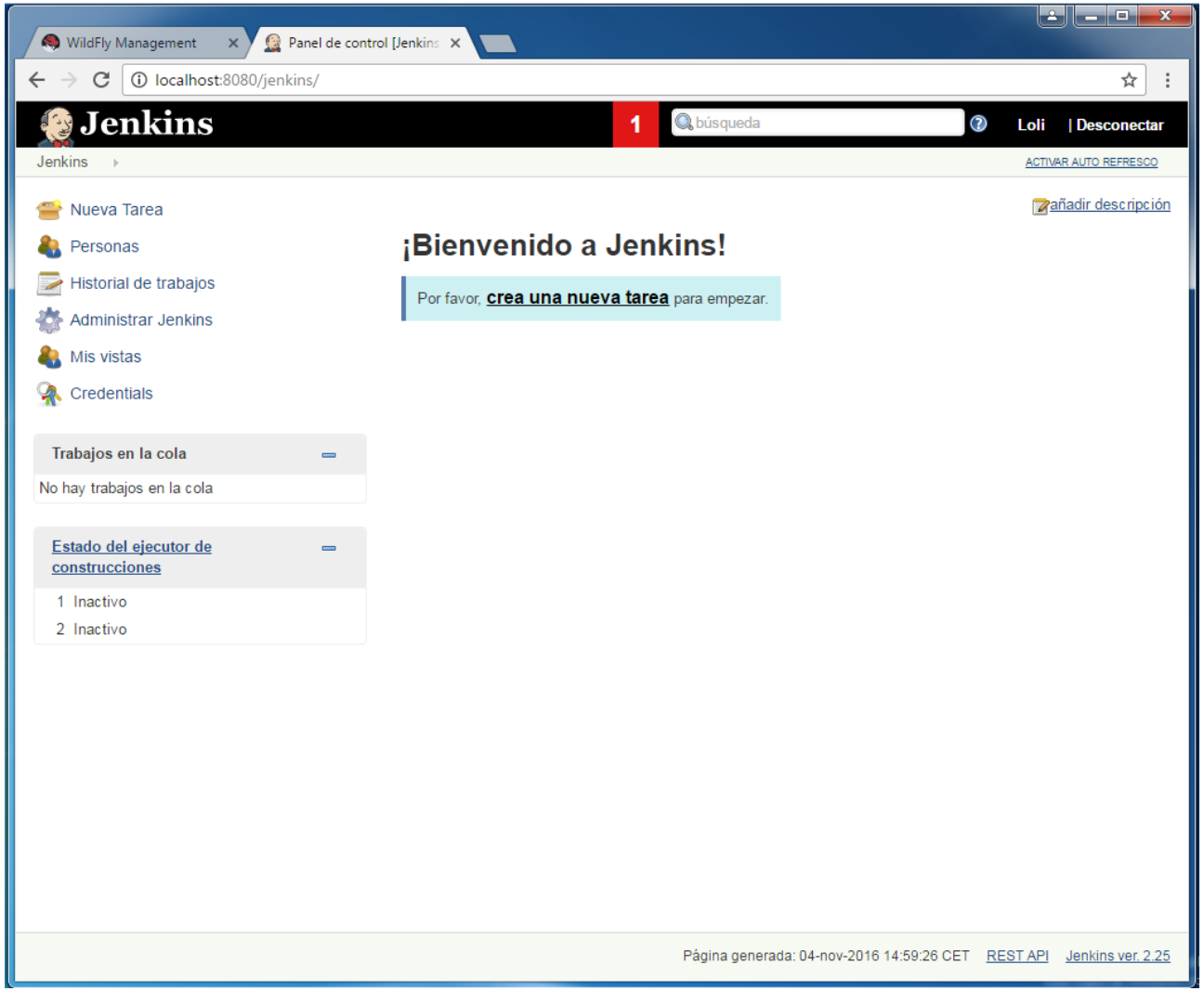
- Pasos para instalar y usar Jenkins:
  - Ejecutar en un servidor (p.ej., Tomcat, Wildfly) o en modo standalone
    - No requiere base de datos
  - Configurar el servidor
  - Añadir usuarios
  - Añadir los credenciales
  - Instalar plugins
  - Añadir tareas

Jenkins

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

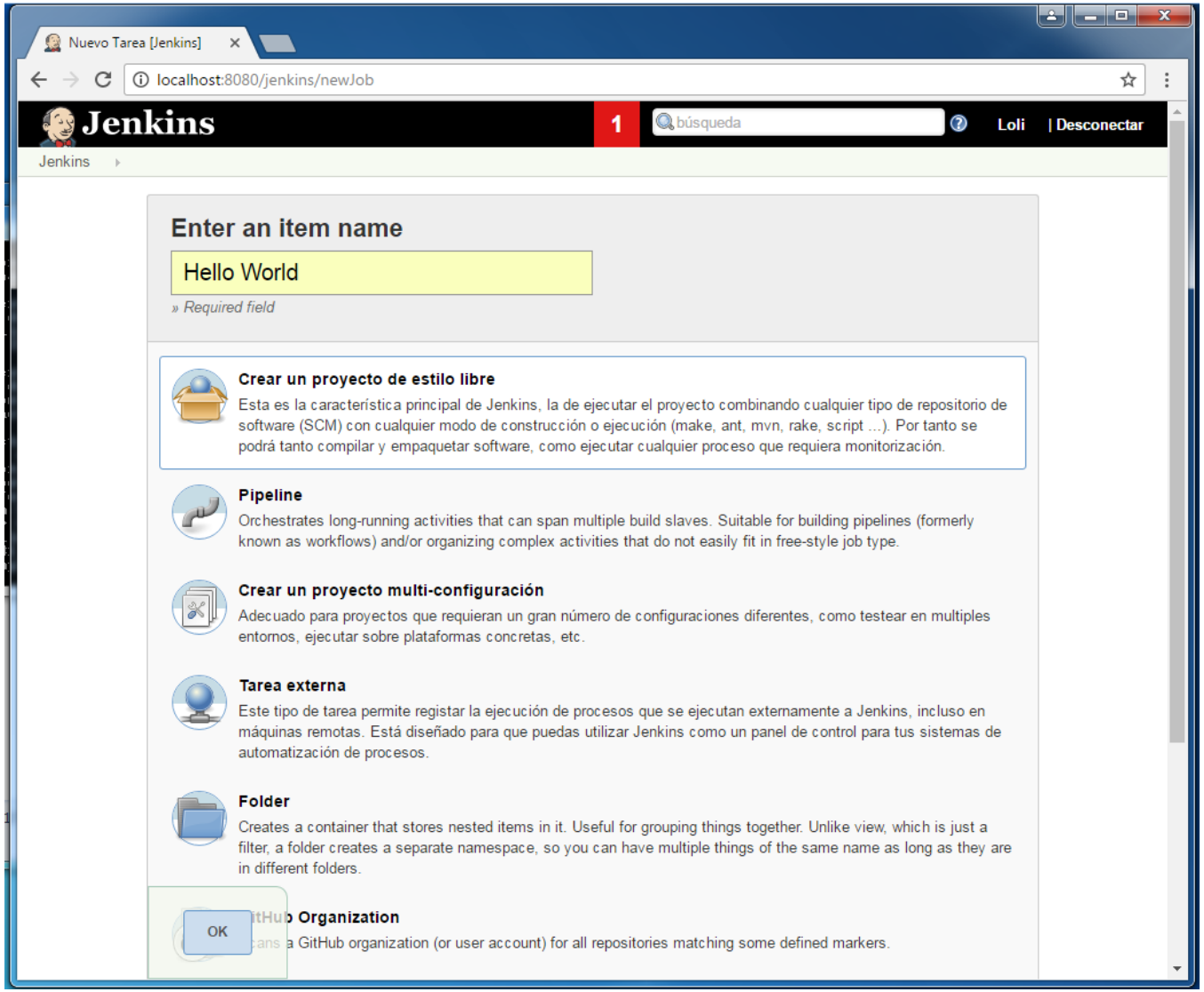


Jenkins


WildFly Management
Panel de control [Jenkins
localhost:8080/jenkins/
Jenkins
1
búsqueda
Loli
| Desconectar
Jenkins
ACTIVAR AUTO REFRESCO
Nueva Tarea
Personas
Historial de trabajos
Administrar Jenkins
Mis vistas
Credentials
añadir descripción
¡Bienvenido a Jenkins!
Por favor, crea una nueva tarea para empezar.
Trabajos en la cola
No hay trabajos en la cola
Estado del ejecutor de construcciones
1 Inactivo
2 Inactivo
Página generada: 04-nov-2016 14:59:26 CET
REST API
Jenkins ver. 2.25

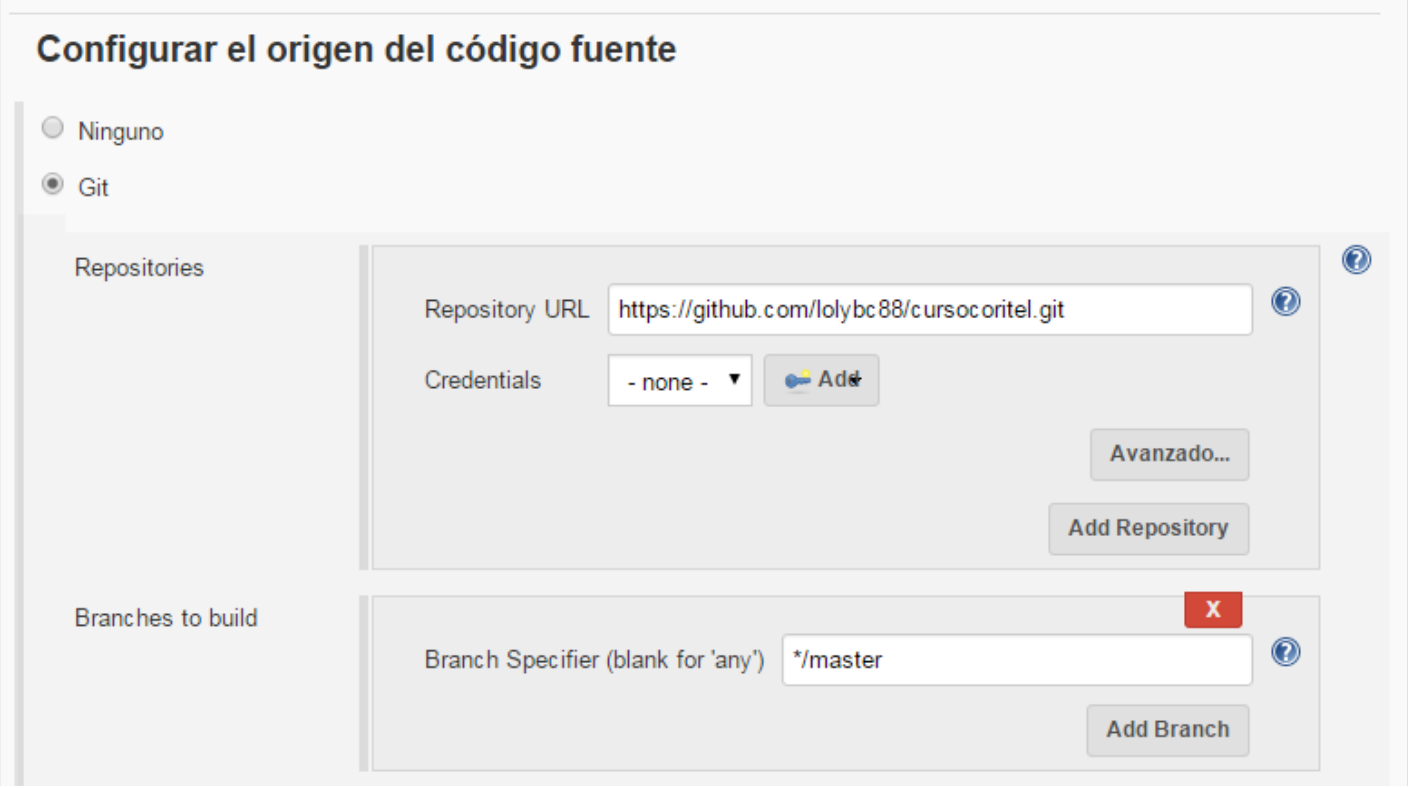
Jenkins

- 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

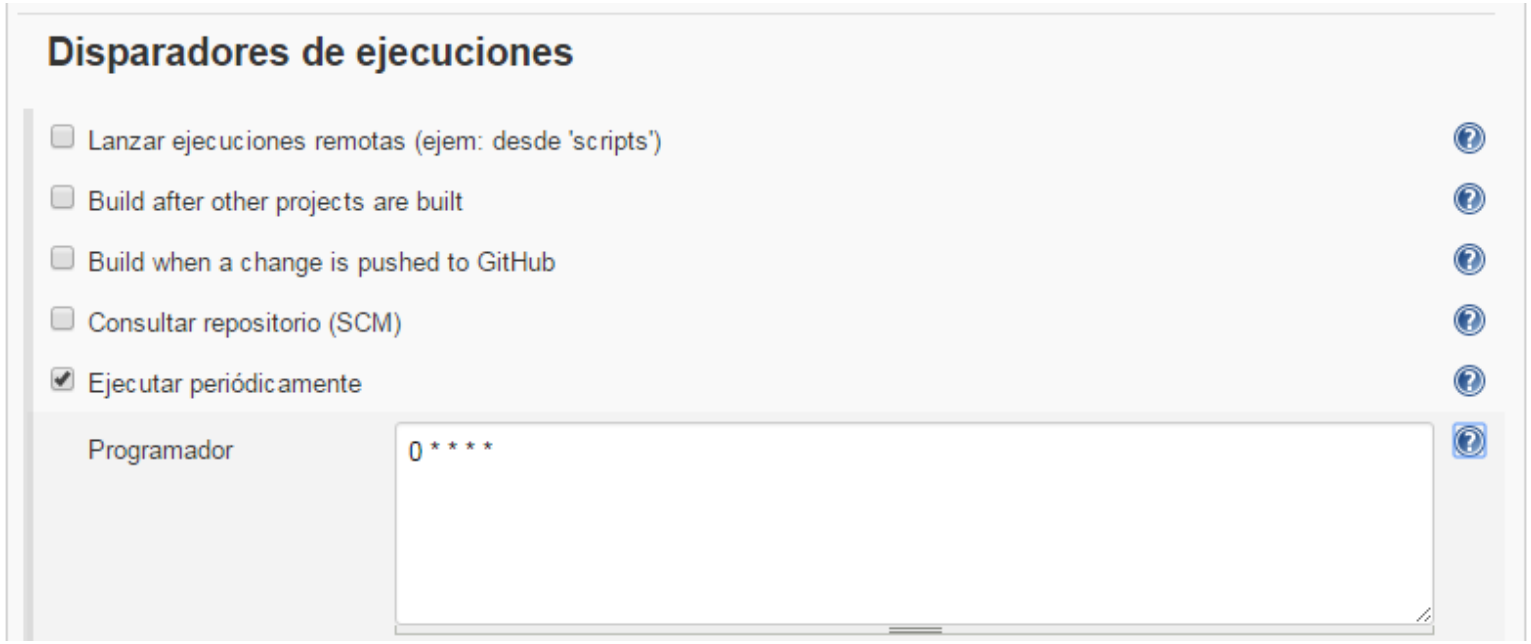


Si no encuentra dónde Git está insalado en el sistema:

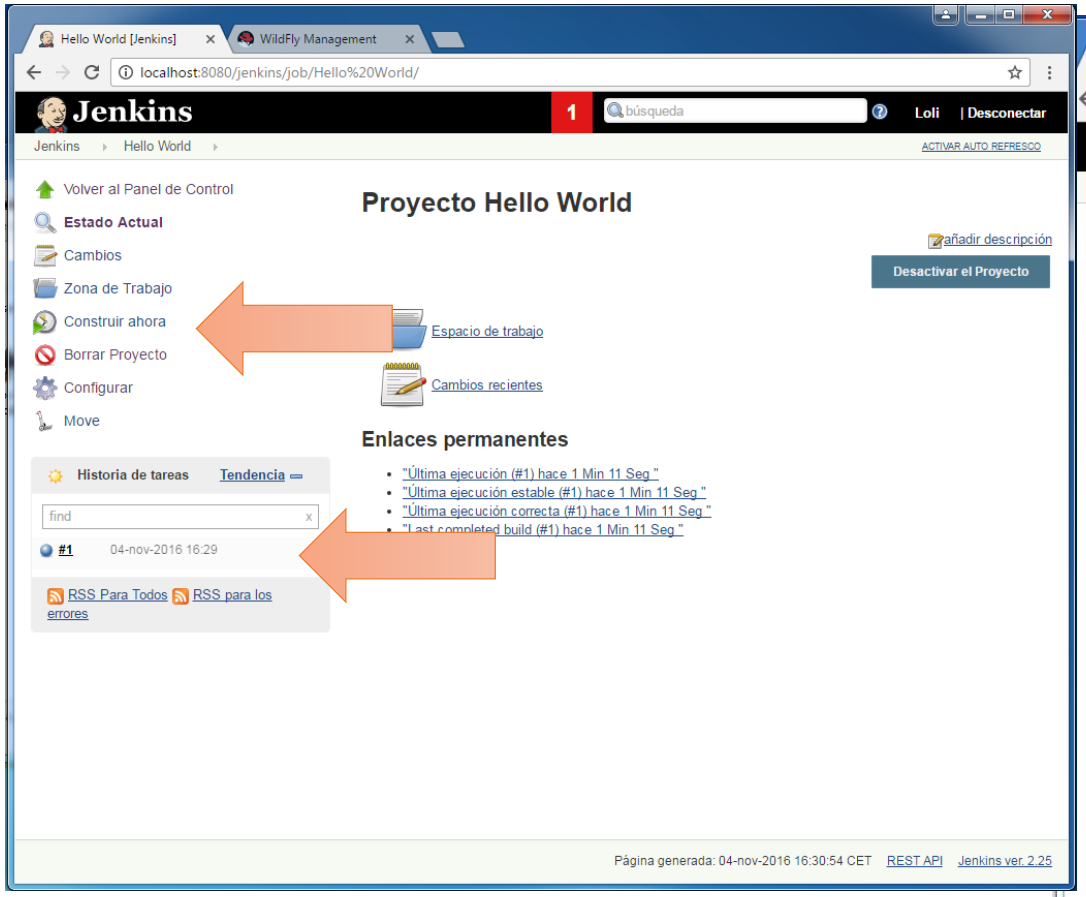
Manage Jenkins > Configure System settings y en la sección Git, cambiar: **Path to Git executable** para que apunte a la ubicación del fichero git.exe

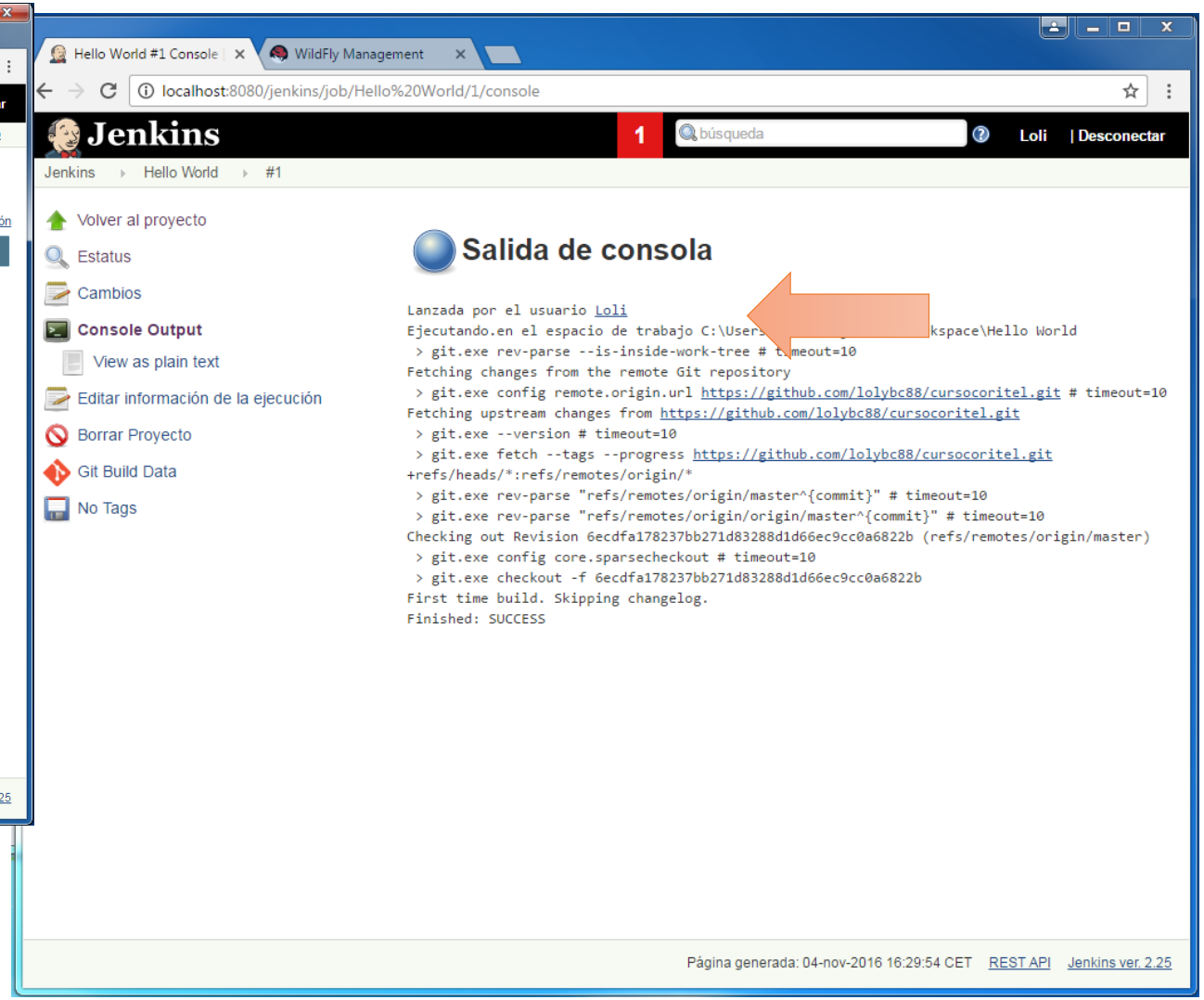
Jenkins

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

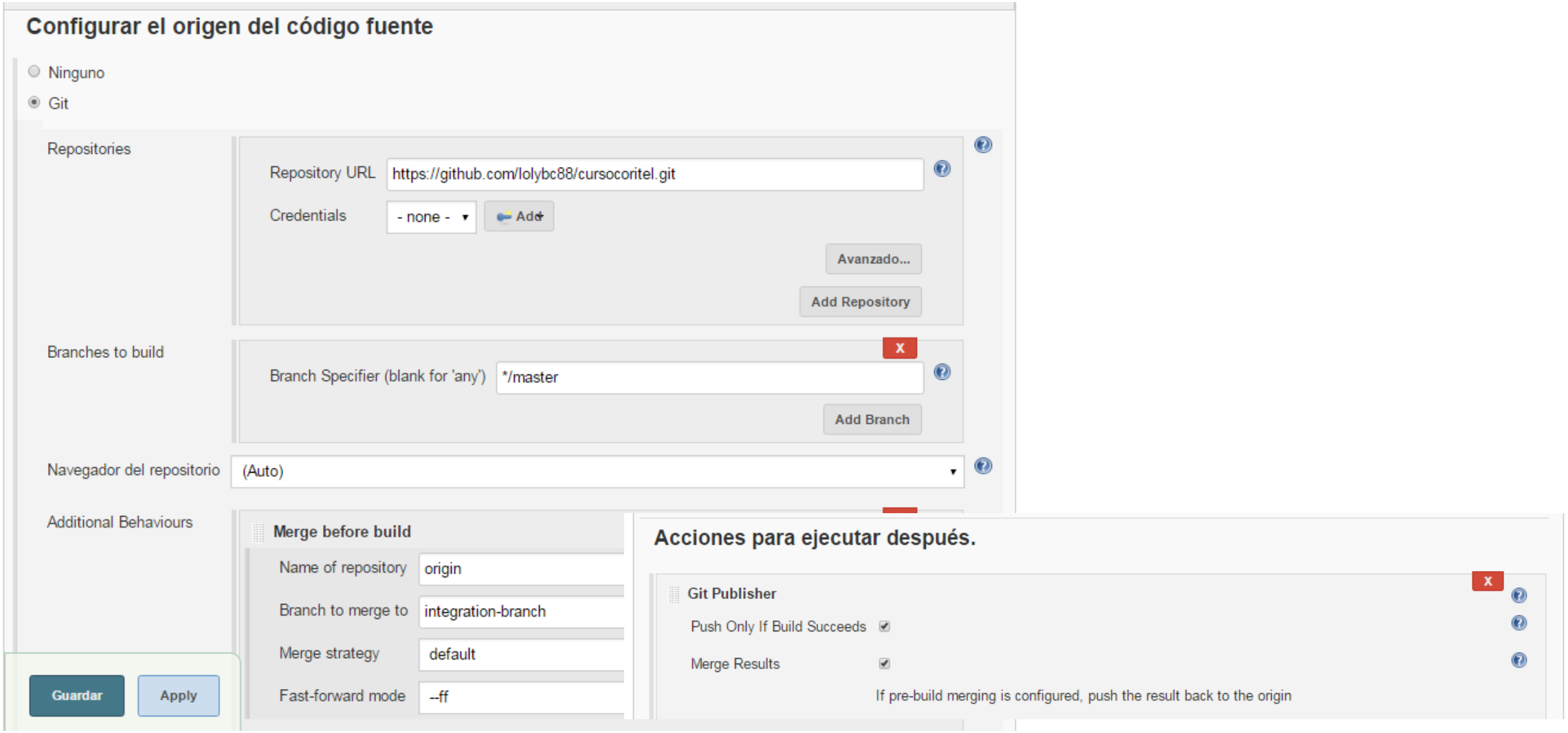


Jenkins


Hello World [Jenkins]
WildFly Management
localhost:8080/jenkins/job/Hello%20World/
Jenkins
Loli | Desconectar
Jenkins ▸ Hello World
ACTIVAR AUTO REFRESCO
Volver al Panel de Control
Estado Actual
Cambios
Zona de Trabajo
Construir ahora
Borrar Proyecto
Configurar
Move
Proyecto Hello World
añadir descripción
Desactivar el Proyecto
Espacio de trabajo
Cambios recientes
Enlaces permanentes
"Última ejecución (#1) hace 1 Min 11 Seg "
"Última ejecución estable (#1) hace 1 Min 11 Seg "
"Última ejecución correcta (#1) hace 1 Min 11 Seg "
Historia de tareas
Tendencia
find
#1
04-nov-2016 16:29
RSS Para Todos
RSS para los errores
Página generada: 04-nov-2016 16:30:54 CET
REST API
Jenkins ver. 2.25


Hello World #1 Console
WildFly Management
localhost:8080/jenkins/job/Hello%20World/1/console
Jenkins
Loli | Desconectar
Jenkins ▸ Hello World ▸ #1
Volver al proyecto
Estatus
Cambios
Console Output
View as plain text
Editar información de la ejecución
Borrar Proyecto
Git Build Data
No Tags
Salida de consola
Lanzada por el usuario Loli
Ejecutando.en el espacio de trabajo C:\Users
kspace\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
Página generada: 04-nov-2016 16:29:54 CET
REST API
Jenkins ver. 2.25

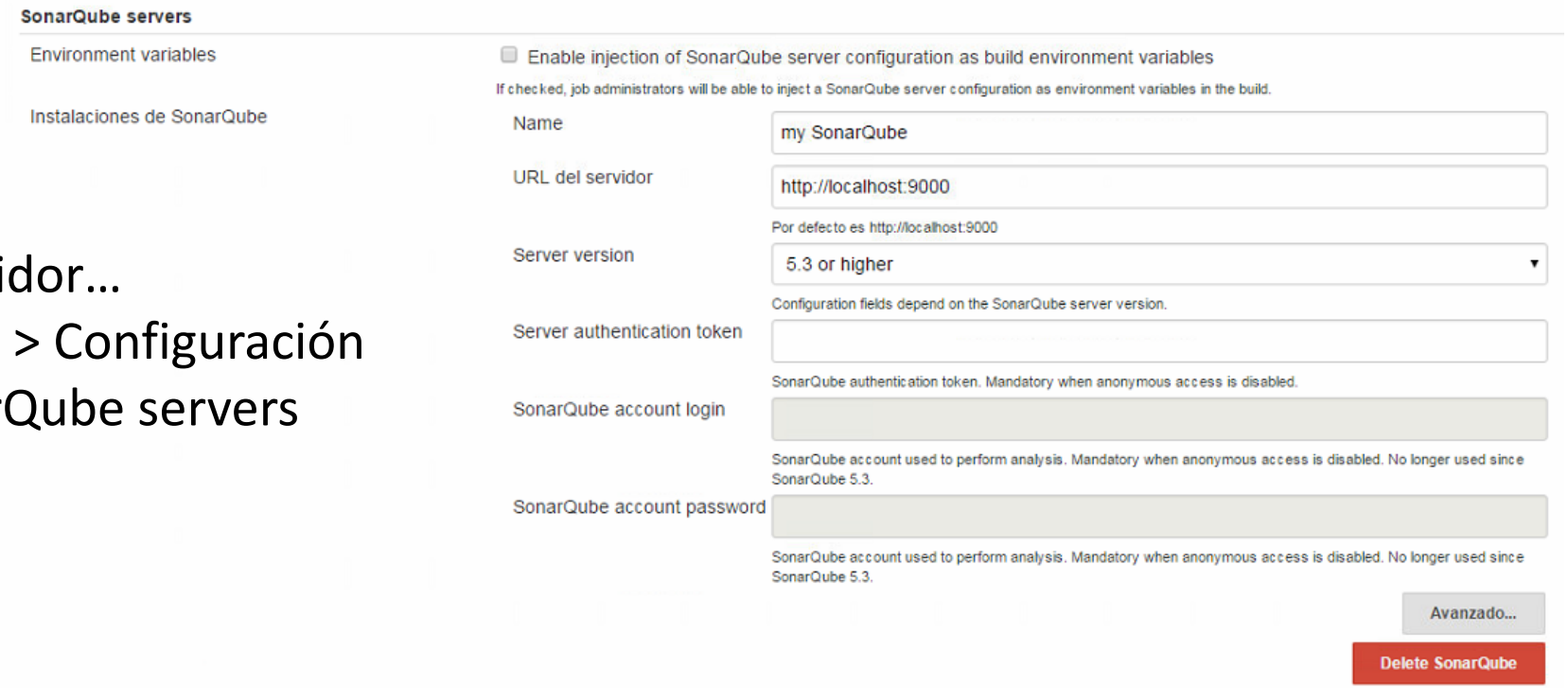
Git y Jenkins



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



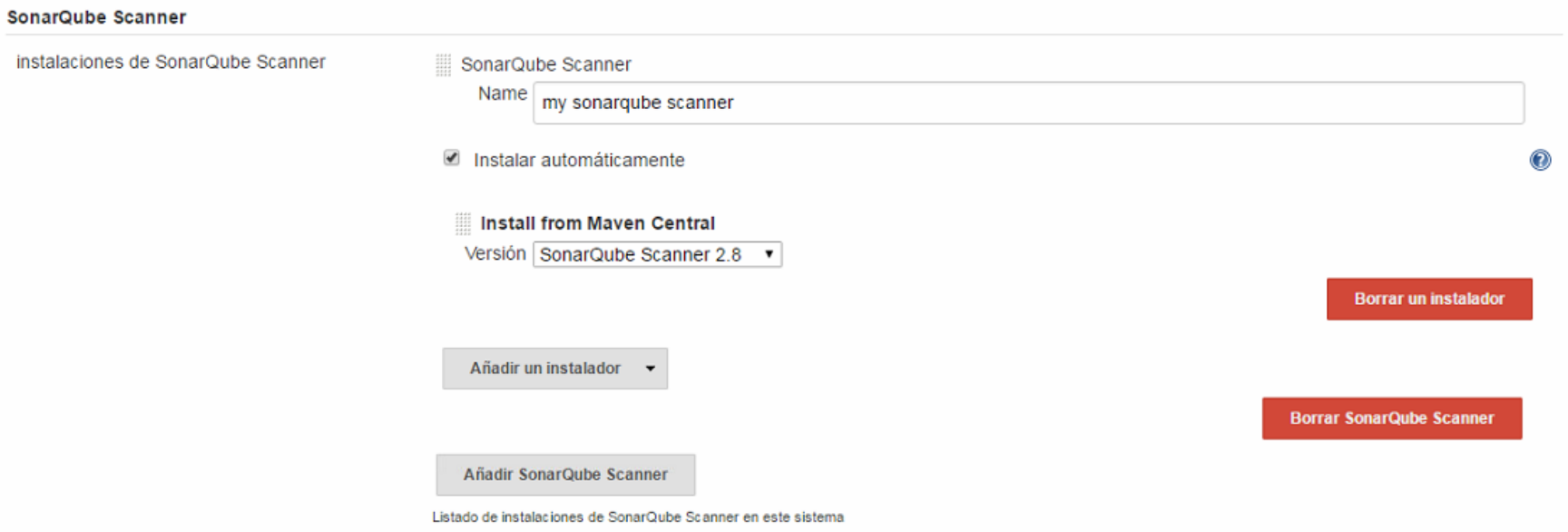
Configuración del servidor…

- Administrar Jenkins > Configuración del Sistema > SonarQube servers

SonarQube y Jenkins

Añadiendo el scanner…

- Administrar Jenkins > Global tool configuration

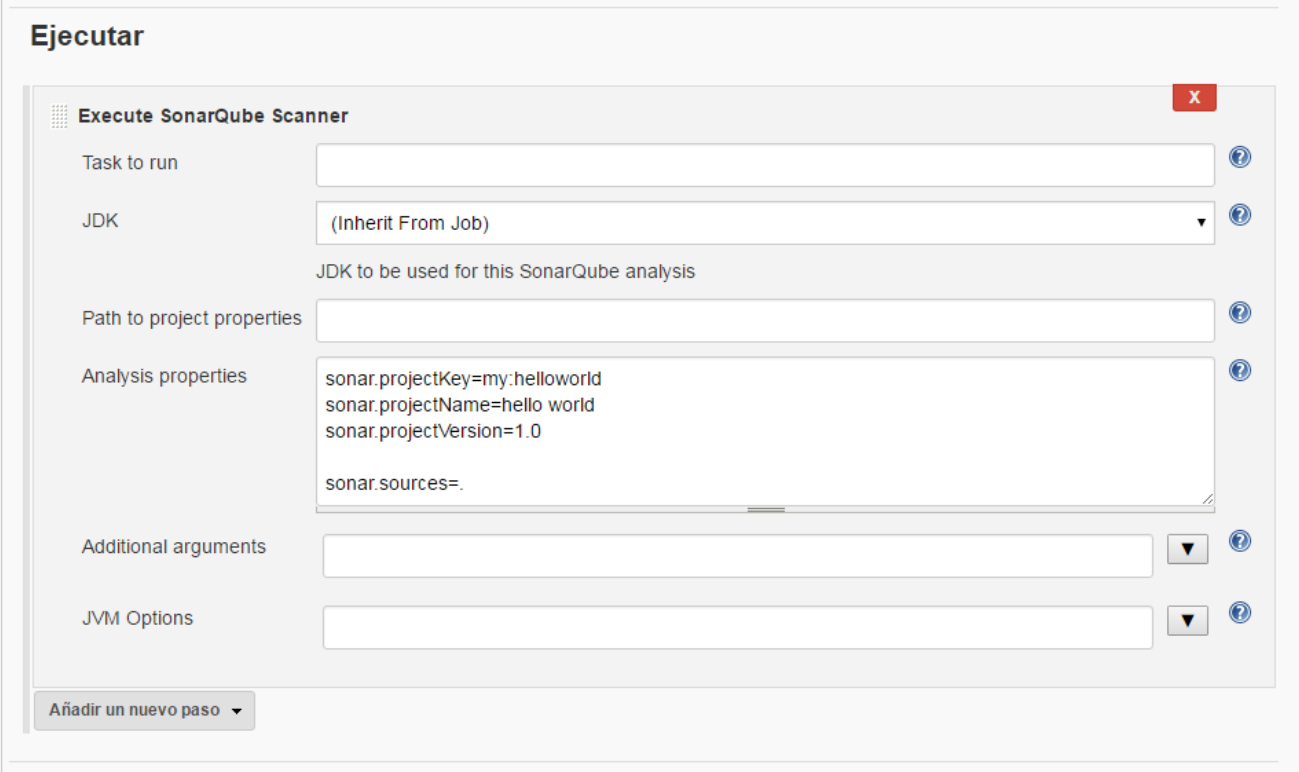


- Podemos escoger una version instalada (deselecionando *'Install automatically'*) o permitir a Jenkins realizer la instalación desde una ubicación remota (seleccionado '*Install automatically*')

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.



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