

## INTEGRACIÓN CONTINUA

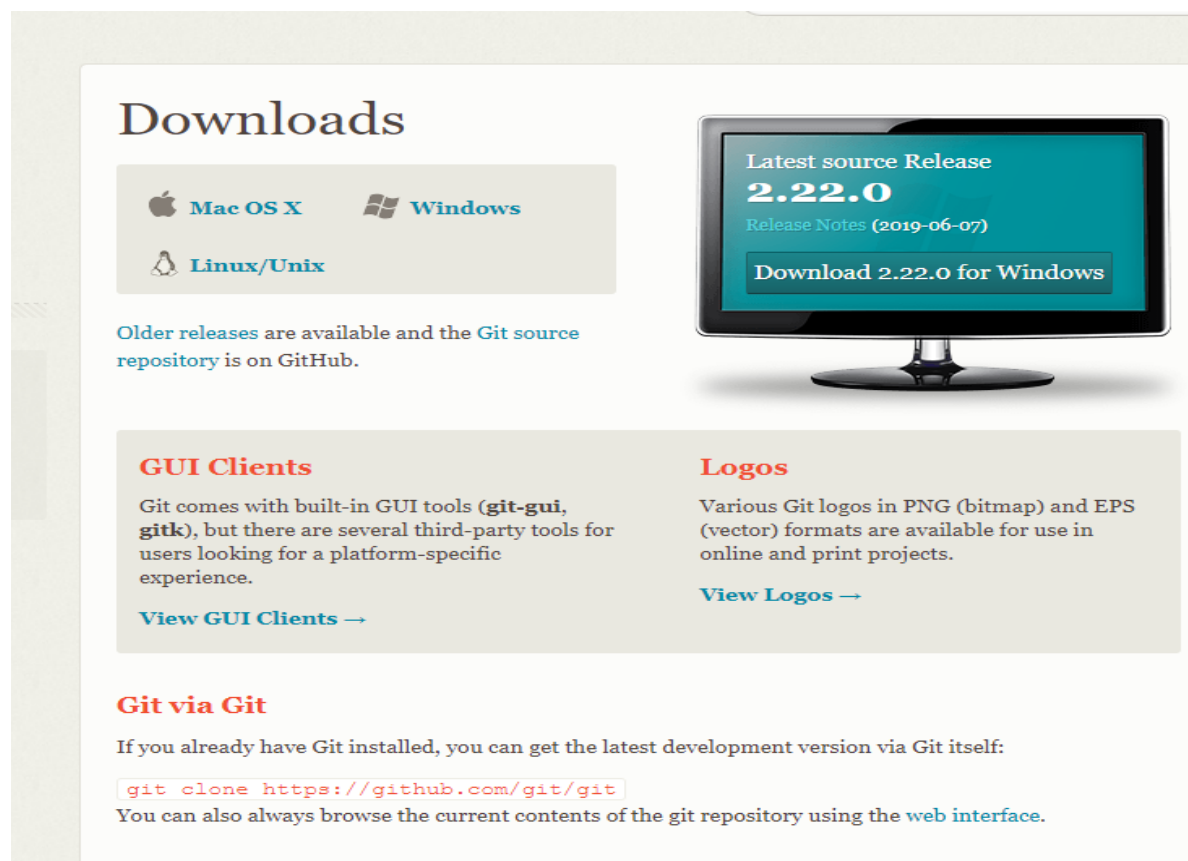
### RESOLUCIÓN DE EJERCICIOS

A continuación encontrará el desarrollo de los ejercicios que resolvió en la semana. Contraste las respuestas entregadas por el docente con las desarrolladas por usted. En caso que no coincidan, y persistan dudas, le invitamos a repasar los contenidos y/ o consultar con su profesor.

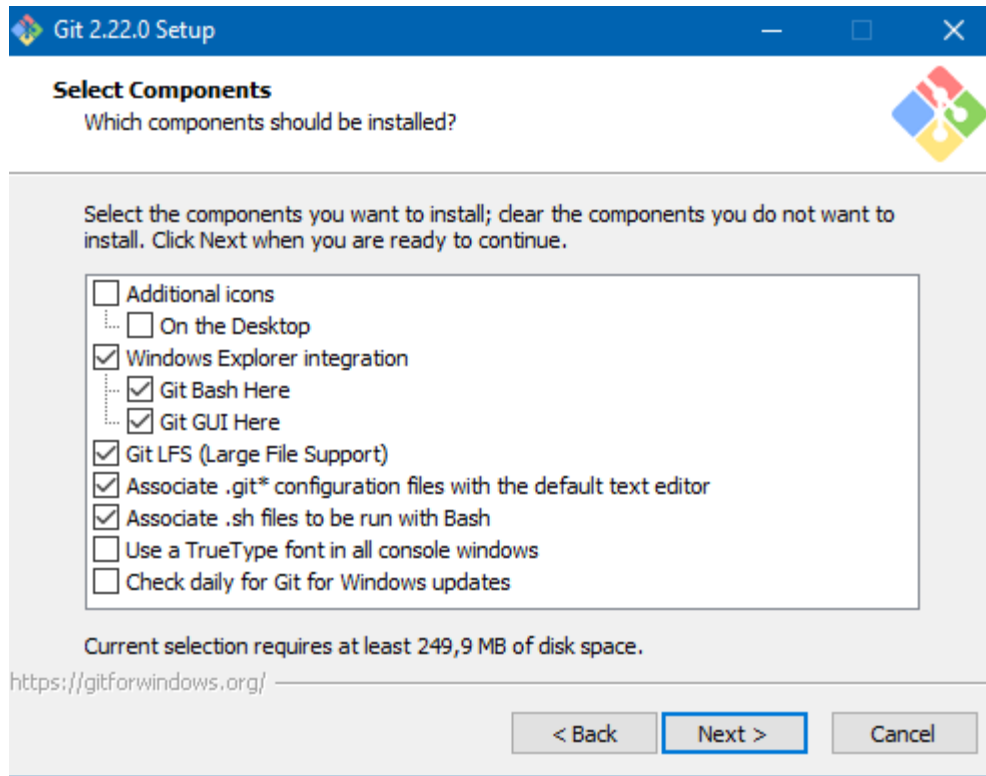
#### 1. Crea un trabajo que indague SCM (GitHub)

Para la creación de un trabajo que indague la herramienta SCM, debemos descargar la herramienta de Git, con esta podremos gestionar nuestro repositorio de forma local para luego poder subir este a el repositorio web de Github.

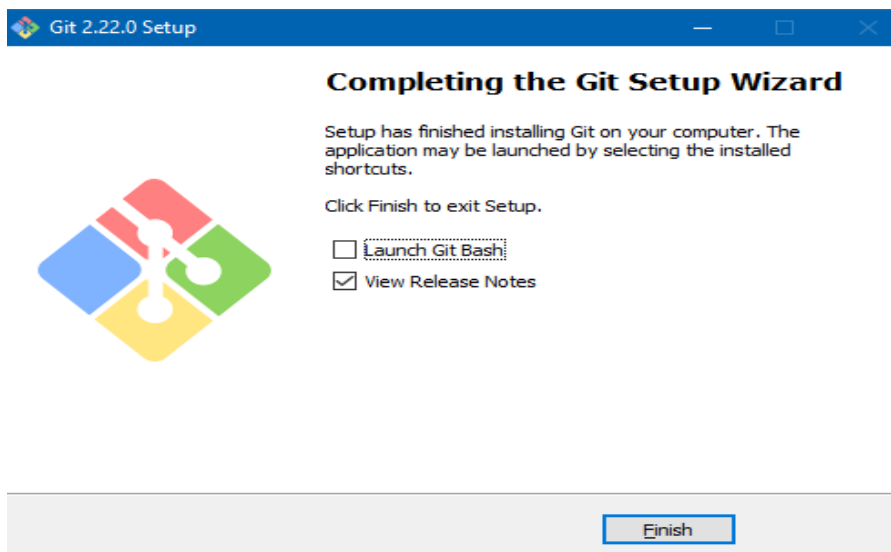
Paso 1: descargar la herramienta Git, nos dirigimos a la página principal de Git <https://git-scm.com/downloads>, luego presionamos el botón de donwload.



## Paso 2: Realizamos la instalación de Git



Dejamos todo por defecto como editor por default VIM y seguimos con el proceso de instalación dejando todo por default hasta que aparezca la pantalla de finalización.



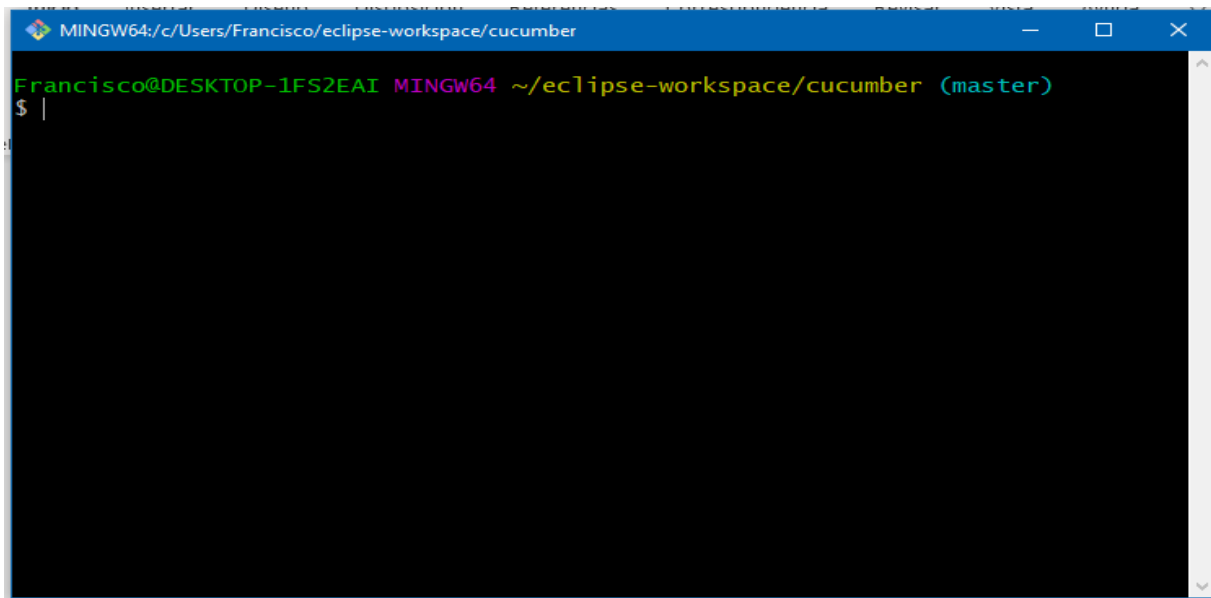
Paso 3: Teniendo Git instalado procederemos a subir nuestro repositorio en GitHub, para eso nos dirigimos a nuestro workspace de eclipse y abrimos la carpeta del proyecto correspondiente(para este ejemplo utilizaremos el proyecto de cucumber), una vez dentro de la carpeta presionamos click derecho y la opción de Git Bash Here.

Este equipo > Disco local (C:) > Usuarios > Francisco > eclipse-workspace > cucumber

Nombre	Fecha de modificación	Tipo	Tamaño
.git	03-08-2019 17:25	Carpeta de archivos	
.settings	30-07-2019 23:20	Carpeta de archivos	
Features	30-07-2019 23:20	Carpeta de archivos	
src	30-07-2019 23:20	Carpeta de archivos	
target	03-08-2019 18:13	Carpeta de archivos	
.classpath	15-07-2019 0:10	Archivo CLASSPATH	2 KB
.project	15-07-2019 0:10	Archivo PROJECT	1 KB
pom	03-08-2019 17:18	Documento XML	2 KB
README.md	31-07-2019 0:11	Archivo MD	0 KB

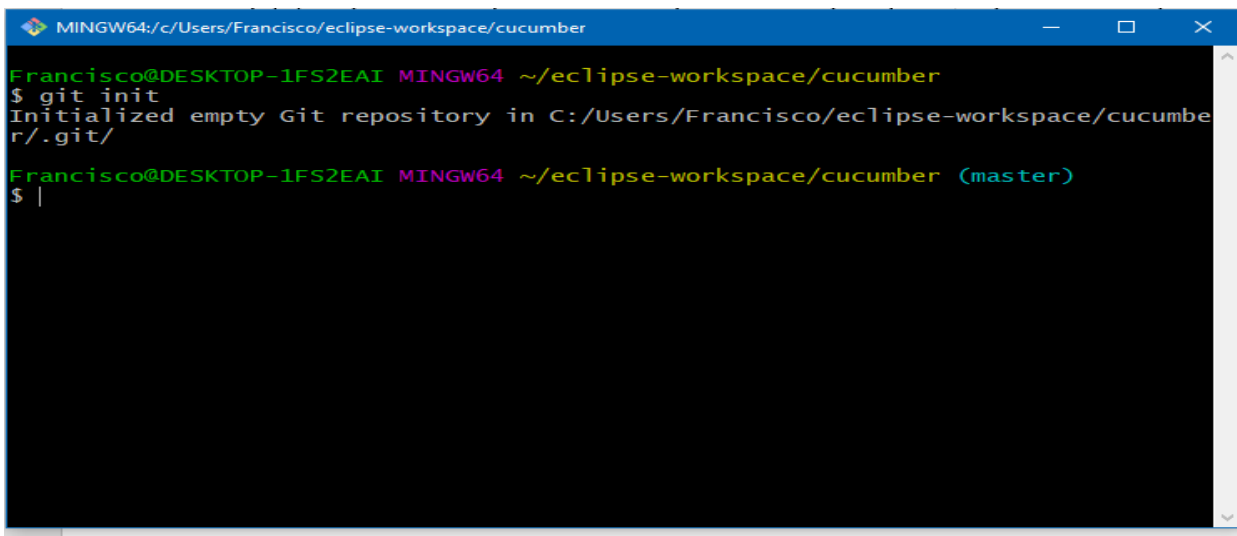
- Ver >
- Ordenar por >
- Agrupar por >
- Actualizar
- Personalizar esta carpeta...
- Pegar
- Pegar acceso directo
- Git GUI Here
- Git Bash Here
- Conceder acceso a >
- Nuevo >
- Propiedades

Al seleccionar opción mencionada se abrirá una consola bash de git como muestra la imagen.



```
MINGW64:/c/Users/Francisco/eclipse-workspace/cucumber
Francisco@DESKTOP-1FS2EAI MINGW64 ~/eclipse-workspace/cucumber (master)
$ |
```

Una vez abierto la consola de git presionaremos el comando git init, lo que iniciara el repositorio git vacio



```
MINGW64:/c/Users/Francisco/eclipse-workspace/cucumber
Francisco@DESKTOP-1FS2EAI MINGW64 ~/eclipse-workspace/cucumber
$ git init
Initialized empty Git repository in C:/Users/Francisco/eclipse-workspace/cucumber/.git/
Francisco@DESKTOP-1FS2EAI MINGW64 ~/eclipse-workspace/cucumber (master)
$ |
```

Luego para preparar nuestro proyecto veremos el estado con el comando `git status` con esto veremos los archivos que están en la carpeta y que no están preparados para subir a nuestro repositorio en color rojo

```
MINGW64:/c/Users/Francisco/eclipse-workspace/cucumber
Francisco@DESKTOP-1FS2EAI MINGW64 ~/eclipse-workspace/cucumber (master)
$ git status
On branch master

No commits yet

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

        .classpath
        .project
        .settings/
        Features/
        README.md
        pom.xml
        src/
        target/

nothing added to commit but untracked files present (use "git add" to track)
Francisco@DESKTOP-1FS2EAI MINGW64 ~/eclipse-workspace/cucumber (master)
$ |
```

Como nos sugiere la consola ingresaremos el comando `git add <file>` con este comando y el parametro solamente añadiremos un archivo, como por ejemplo el archivo `.classpath`

```
MINGW64:/c/Users/Francisco/eclipse-workspace/cucumber
$ git status
On branch master

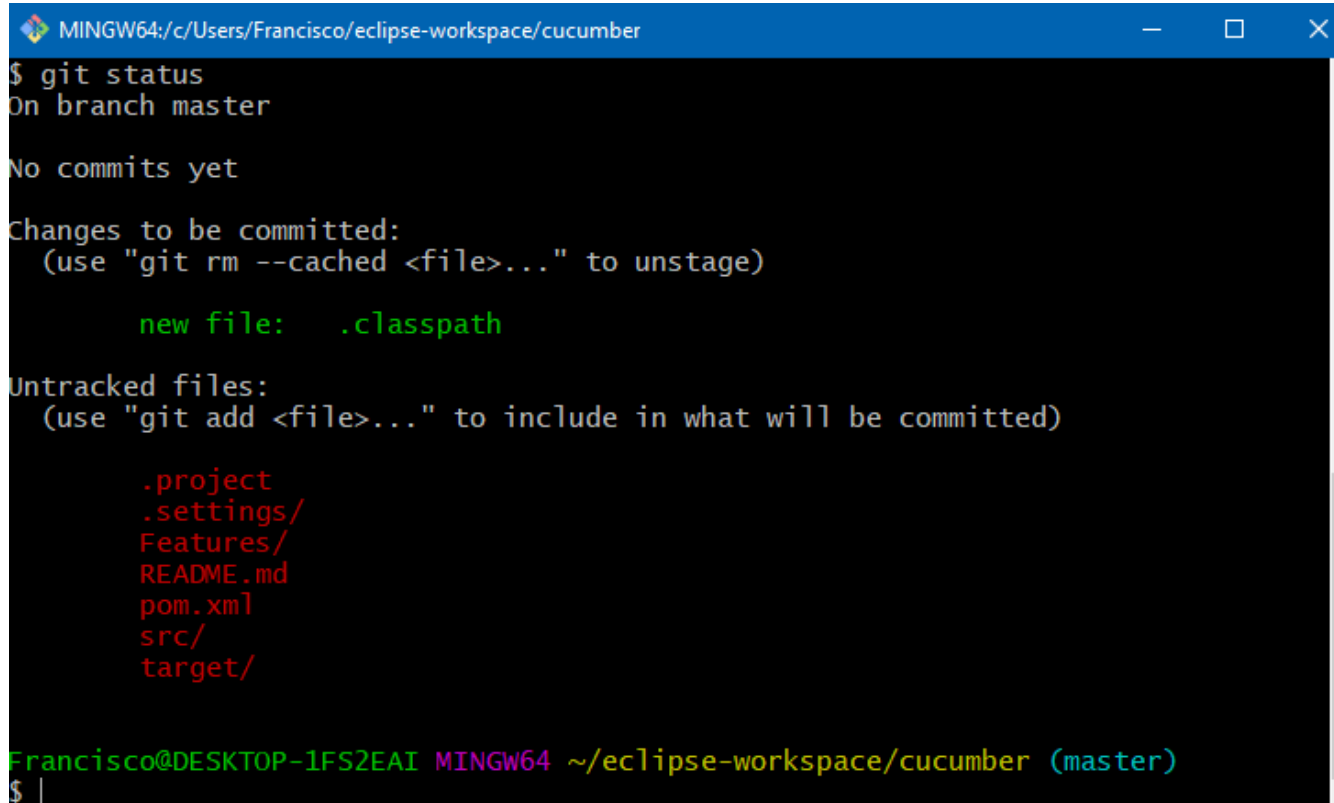
No commits yet

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

        .classpath
        .project
        .settings/
        Features/
        README.md
        pom.xml
        src/
        target/

nothing added to commit but untracked files present (use "git add" to track)
Francisco@DESKTOP-1FS2EAI MINGW64 ~/eclipse-workspace/cucumber (master)
$ git add .classpath
Francisco@DESKTOP-1FS2EAI MINGW64 ~/eclipse-workspace/cucumber (master)
$ |
```

Nos indica que se ha añadido el archivo .classpath, para verificar presionaremos el comando git status y nos indicara que esta preparado para realizar el commit



```
MINGW64:/c/Users/Francisco/eclipse-workspace/cucumber
$ git status
On branch master

No commits yet

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

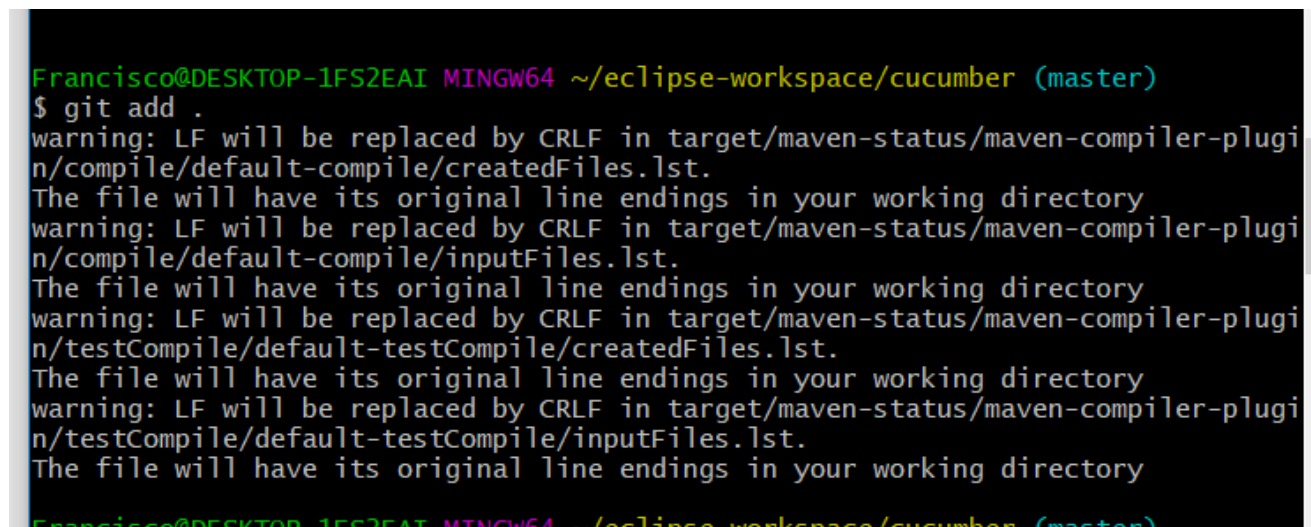
        new file:   .classpath

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

        .project
        .settings/
        Features/
        README.md
        pom.xml
        src/
        target/

Francisco@DESKTOP-1FS2EAI MINGW64 ~/eclipse-workspace/cucumber (master)
$ |
```

Ahora agregaremos todos los archivos de el directorio en donde se encuentra nuestra el proyecto con el comando git add .



```
Francisco@DESKTOP-1FS2EAI MINGW64 ~/eclipse-workspace/cucumber (master)
$ git add .
warning: LF will be replaced by CRLF in target/maven-status/maven-compiler-plugin/compile/default-compile/createdFiles.lst.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in target/maven-status/maven-compiler-plugin/compile/default-compile/inputFiles.lst.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in target/maven-status/maven-compiler-plugin/testCompile/default-testCompile/createdFiles.lst.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in target/maven-status/maven-compiler-plugin/testCompile/default-testCompile/inputFiles.lst.
The file will have its original line endings in your working directory

Francisco@DESKTOP-1FS2EAI MINGW64 ~/eclipse-workspace/cucumber (master)
```

```

MINGW64:/c/Users/Francisco/eclipse-workspace/cucumber
bash: status: command not found

Francisco@DESKTOP-1FS2EAI MINGW64 ~/eclipse-workspace/cucumber (master)
$ git status
On branch master

No commits yet

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

        new file:   .classpath
        new file:   .project
        new file:   .settings/org.eclipse.core.resources.prefs
        new file:   .settings/org.eclipse.jdt.core.prefs
        new file:   .settings/org.eclipse.m2e.core.prefs
        new file:   Features/MyTest.feature
        new file:   README.md
        new file:   pom.xml
        new file:   src/main/java/com/iplacex/cucumber/App.java
        new file:   src/test/java/StepDefinition/Steps.java
        new file:   src/test/java/TestRunner/Runner.java
        new file:   target/classes/META-INF/MANIFEST.MF
        new file:   target/classes/META-INF/maven/com.iplacex/cucumber/pom.prope


```

al consultar el status de nuestros archivos nos mostrara de color verde de que están preparados. Pausaremos el proceso de la consola para crear nuestro repositorio en el sitio GitHub, ingresamos al sitio <https://github.com/>, nos registramos y agregaremos un nuevo repositorio semana 6

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?  
[Import a repository.](#)

Owner

 FranciscoGallardo3180 ▾

Repository name \*

/ semana6 ✓

Great repository names are short and memorable. Need inspiration? How about **upgraded-eureka**?

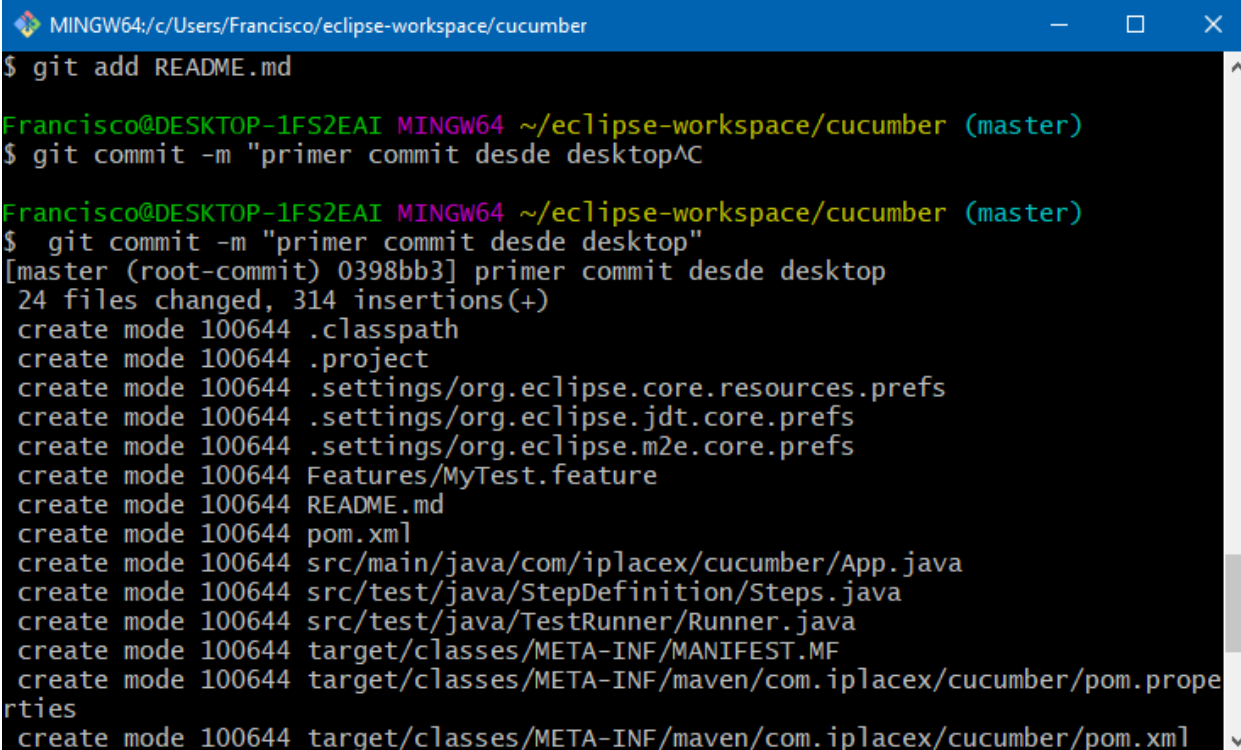
Description (optional)

Una vez que hayamos creado nuestro repositorio en GitHub, nos mostrara unas instrucciones para agregar por consola de Git, como ya hemos adelantado ingresando los primeros comandos, seguimos desde el comando git add readme.md

### ...or create a new repository on the command line

```
echo "# semana6" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/FranciscoGallardo3180/semana6.git
git push -u origin master
```

Nos dirigimos a nuestra consola de git e ingresamos los comandos que nos indica el sitio crearemos nuestro primer commit presinando git commit -m "primer commit desde desktop"



```
MINGW64; c:/Users/Francisco/eclipse-workspace/cucumber
$ git add README.md
Francisco@DESKTOP-1FS2EAI MINGW64 ~/eclipse-workspace/cucumber (master)
$ git commit -m "primer commit desde desktop"
Francisco@DESKTOP-1FS2EAI MINGW64 ~/eclipse-workspace/cucumber (master)
$ git commit -m "primer commit desde desktop"
[master (root-commit) 0398bb3] primer commit desde desktop
24 files changed, 314 insertions(+)
create mode 100644 .classpath
create mode 100644 .project
create mode 100644 .settings/org.eclipse.core.resources.prefs
create mode 100644 .settings/org.eclipse.jdt.core.prefs
create mode 100644 .settings/org.eclipse.m2e.core.prefs
create mode 100644 Features/MyTest.feature
create mode 100644 README.md
create mode 100644 pom.xml
create mode 100644 src/main/java/com/iplacex/cucumber/App.java
create mode 100644 src/test/java/StepDefinition/Steps.java
create mode 100644 src/test/java/TestRunner/Runner.java
create mode 100644 target/classes/META-INF/MANIFEST.MF
create mode 100644 target/classes/META-INF/maven/com.iplacex/cucumber/pom.properties
create mode 100644 target/classes/META-INF/maven/com.iplacex/cucumber/pom.xml
```



Luego añadiremos el servidor remoto el cual se comunicará con nuestro repositorio de git de forma local

```
Francisco@DESKTOP-1FS2EAI MINGW64 ~/eclipse-workspace/cucumber (master)
$ git remote add origin https://github.com/FranciscoGallardo3180/semana6.git

Francisco@DESKTOP-1FS2EAI MINGW64 ~/eclipse-workspace/cucumber (master)
$ |
```

Finalmente subiremos nuestro proyecto a un repositorio remoto con el comando `git push -u origin master`

Nos pedirá iniciar sesión de nuestra cuenta de GitHub, ingresamos las credenciales y nos mostrara la siguiente pantalla.

```
Francisco@DESKTOP-1FS2EAI MINGW64 ~/eclipse-workspace/cucumber (master)
$ git push -u origin master
Enumerating objects: 56, done.
Counting objects: 100% (56/56), done.
Delta compression using up to 12 threads
Compressing objects: 100% (34/34), done.
Writing objects: 100% (56/56), 9.41 KiB | 1.57 MiB/s, done.
Total 56 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/FranciscoGallardo3180/semana6.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

Francisco@DESKTOP-1FS2EAI MINGW64 ~/eclipse-workspace/cucumber (master)
$ |
```

Verificamos que se haya subido todo en el repositorio y veremos una pantalla como la siguiente

FranciscoGallardo3180 / semana6

Unwatch 1 Star 0 Fork 0

Code

Issues 0

Pull requests 0

Projects 0

Wiki

Security

Insights

Settings

No description, website, or topics provided.

Manage topics

1 commit

1 branch

0 releases

1 contributor

Branch: master

New pull request

Create new file

Upload files

Find File

Clone or download

FranciscoGallardo3180 primer commit desde desktop

Latest commit 0398bb3 6 minutes ago

.settings	primer commit desde desktop	6 minutes ago
Features	primer commit desde desktop	6 minutes ago
src	primer commit desde desktop	6 minutes ago
target	primer commit desde desktop	6 minutes ago
.classpath	primer commit desde desktop	6 minutes ago
.project	primer commit desde desktop	6 minutes ago
README.md	primer commit desde desktop	6 minutes ago
pom.xml	primer commit desde desktop	6 minutes ago

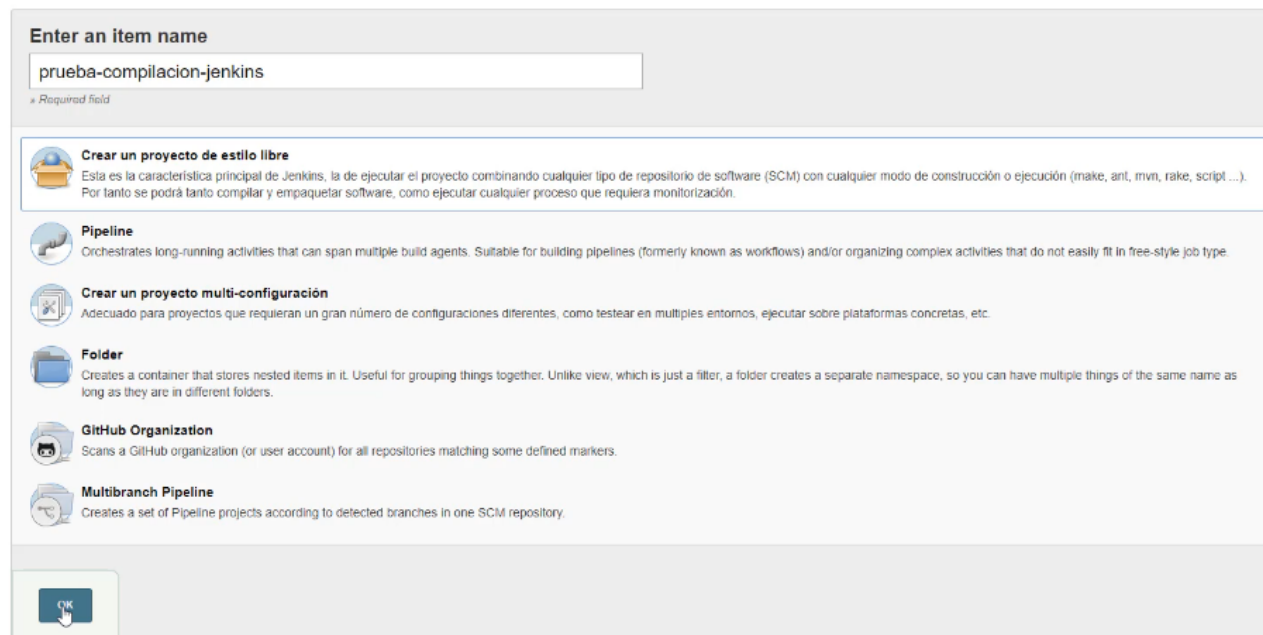
Help people interested in this repository understand your project by adding a README.

Add a README

2. Crear un trabajo que realice la compilación y pruebas desde repositorio GitHub con Jenkins

Como ya tenemos nuestro repositorio listo para ser consumido por nuestro servidor de integración de Jenkins, realizaremos la creación de una nueva tarea que realiza la compilación y pureba de nuestro código almacenado en GitHub.







Paso 1: Nos dirigimos a nuestro servidor de Integración continua Jenkins y crearemos una tarea de estilo libre.



Enter an item name

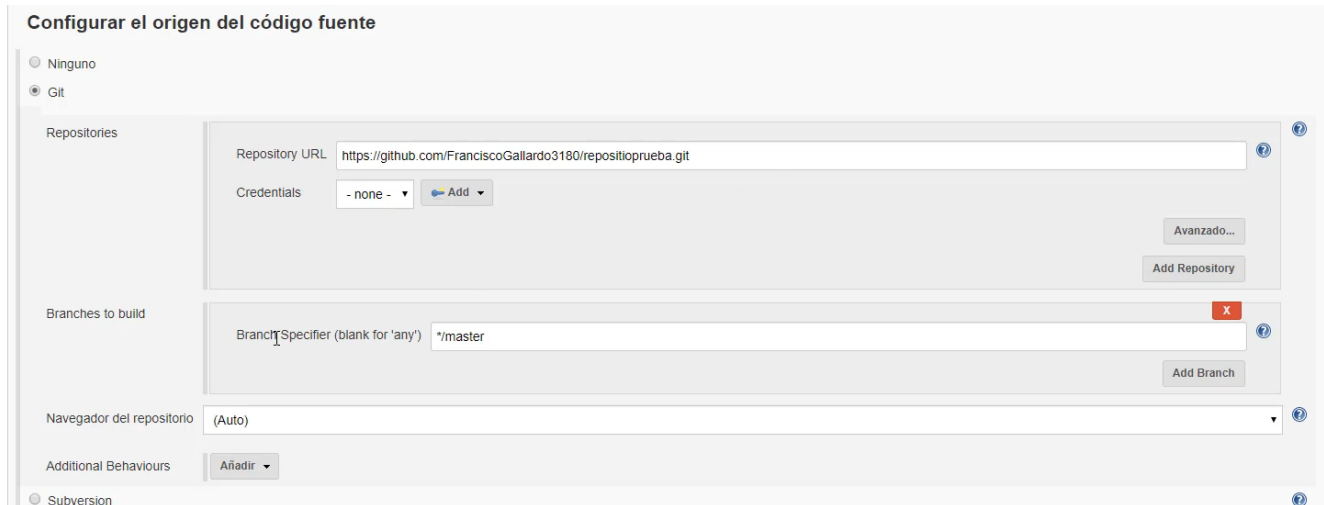
prueba-compilacion-jenkins

\* Required field

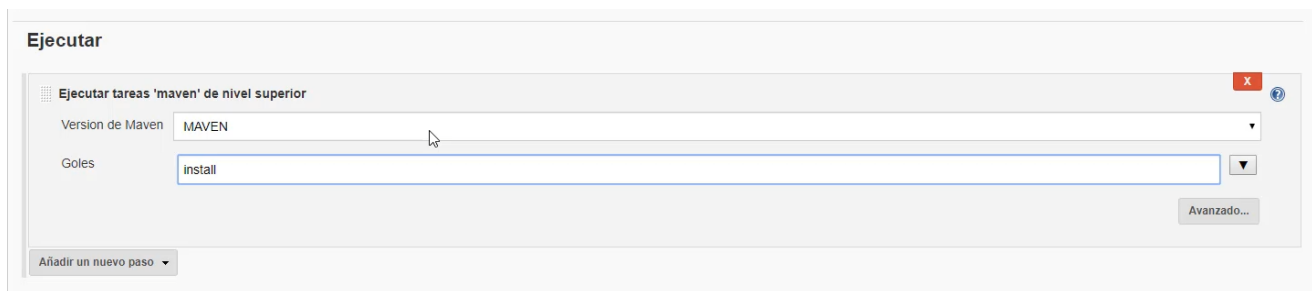
-  **Crear un proyecto de estilo libre**  
Esta es la característica principal de Jenkins, la de ejecutar el proyecto combinando cualquier tipo de repositorio de software (SCM) con cualquier modo de construcción o ejecución (make, ant, mvn, rake, script ...). Por tanto se podrá tanto compilar y empaquetar software, como ejecutar cualquier proceso que requiera monitorización.
-  **Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
-  **Crear un proyecto multi-configuración**  
Adecuado para proyectos que requieran un gran número de configuraciones diferentes, como testear en multiples entornos, ejecutar sobre plataformas concretas, etc.
-  **Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
-  **GitHub Organization**  
Scans a GitHub organization (or user account) for all repositories matching some defined markers.
-  **Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK

Luego en la configuración de nuestro proyecto nos dirigimos a la sección de configuración de origen de código fuente en donde uniremos el repositorio alojado en el sito de GitHub.



Luego en entorno de ejecuion seleccionamos ejecutar tareas de Maven de nivel superior e ingresamos el Goal de Maven.



Posteriormente realizaremos programación mediante con de nuestra tarea programada para que se ejecute cada 5 minutos.

Nota: tienen que ser cuidadoso con los espacios en el programador de Jenkins ya que un espacio de mas provoca un error de escritura.

### Disparadores de ejecuciones

- ☐ Lanzar ejecuciones remotas (ejem: desde 'scripts')
- ☐ Construir tras otros proyectos
- ☐ Consultar repositorio (SCM)
- ☒ Ejecutar periódicamente

Programador

`*/* * * * *`

**⚠ Spread load evenly by using `'H/5 * * * *'` rather than `'/5 * * * *'`**  
 Would last have run at sábado 3 de agosto de 2019 00:35:47 Hora de Chile; would next run at sábado 3 de agosto de 2019 00:35:47 Hora de Chile.

This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:

MINUTE	HOUR	DOM	MONTH	DOW
MINUTE	Minutes within the hour (0–59)			
HOUR	The hour of the day (0–23)			
DOM	The day of the month (1–31)			
MONTH	The month (1–12)			

Finalmente realizamos la prueba de nuestra tarea realizando la compilación de nuestro proyecto alojado en GitHub, con los resultados en pantalla.

- Volver al proyecto
- Estatus
- Cambios
- Console Output**
- View as plain text
- Editar información de la ejecución
- Delete build '#1'
- Git Build Data
- No Tags

### Salida de consola

Lanzada por el usuario [Francisco](#)

Running as SYSTEM

Ejecutando en el espacio de trabajo C:\Users\Francisco\.jenkins\workspace\prueba-compilacion-jenkins

No credentials specified

Cloning the remote Git repository

Cloning repository <https://github.com/FranciscoGallardo3180/repositioprueba.git>

> git.exe init C:\Users\Francisco\.jenkins\workspace\prueba-compilacion-jenkins # timeout=10

Fetching upstream changes from <https://github.com/FranciscoGallardo3180/repositioprueba.git>

> git.exe --version # timeout=10

> git.exe fetch --tags --force --progress <https://github.com/FranciscoGallardo3180/repositioprueba.git> +refs/heads/\*:refs/remotes/origin/\* # timeout=10

> git.exe config remote.origin.url <https://github.com/FranciscoGallardo3180/repositioprueba.git> # timeout=10

> git.exe config remote.origin.fetch <https://github.com/FranciscoGallardo3180/repositioprueba.git> # timeout=10

Fetching upstream changes from <https://github.com/FranciscoGallardo3180/repositioprueba.git>

> git.exe fetch --tags --force --progress <https://github.com/FranciscoGallardo3180/repositioprueba.git> +refs/heads/\*:refs/remotes/origin/\* # timeout=10

> git.exe rev-parse "refs/remotes/origin/master^(commit)" # timeout=10

> git.exe rev-parse "refs/remotes/origin/master^(commit)" # timeout=10

Checking out Revision 129b57691cf5f4883f42998fc447e0a82db8fb89 (refs/remotes/origin/master)

> git.exe config core.sparsecheckout # timeout=10

> git.exe checkout -f 129b57691cf5f4883f42998fc447e0a82db8fb89

Commit message: "primer commit desde equipo local"

First time build. Skipping changelog.

[prueba-compilacion-jenkins] \$ cmd.exe /C "C:\dev\apache-maven-3.6.1\bin\mvn.cmd install && exit %ERRORLEVEL%"

[INFO] Scanning for projects...

[INFO]

[INFO] -----c com.iplace:cucumber>-----

[INFO] Building cucumber 0.0.1-SNAPSHOT

[INFO] -----[ jar ]-----


[INFO]


### 3. Crea un trabajo que utilice Jenkinsfile


Para este propósito realizaremos la creación de una tarea pero a diferencia de las anteriores crearemos directamente un pipeline el que mediante scripting realizaremos las tareas como etapas, como primera parte en la tarea a modo de ejemplo crearemos un script sencillo que nos imprima sobre la tarea que esta realizando.


Paso 1: creamos una nueva tarea de tipo Pipeline.


**Enter an item name**  
  
» Required field


**Crear un proyecto de estilo libre**  
Esta es la característica principal de Jenkins, la de ejecutar el proyecto combinando cualquier tipo de repositorio de software (SCM) con cualquier modo de construcción o ejecución (make, ant, mvn, rake, script ...). Por tanto se podrá tanto compilar y empaquetar software, como ejecutar cualquier proceso que requiera monitorización.

**Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Crear un proyecto multi-configuración**  
Adecuado para proyectos que requieran un gran número de configuraciones diferentes, como testear en multiples entornos, ejecutar sobre plataformas concretas, etc.

**Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**GitHub Organization**  
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

**Multi-branch Pipeline**

Paso 2: luego nos dirigimos a pipeline y seleccionamos la definición de Pipeline Script, lo que haremos en este ejemplo será imprimir en primera instancia un Hola mundo, luego dividiremos, por las siguientes etapas ingresando stage('nombre de la etapa'), imprimiendo que esta dentro de la etapa correspondiente.

Presionamos en guardar y realizamos la ejecución.

## Pipeline

Definition

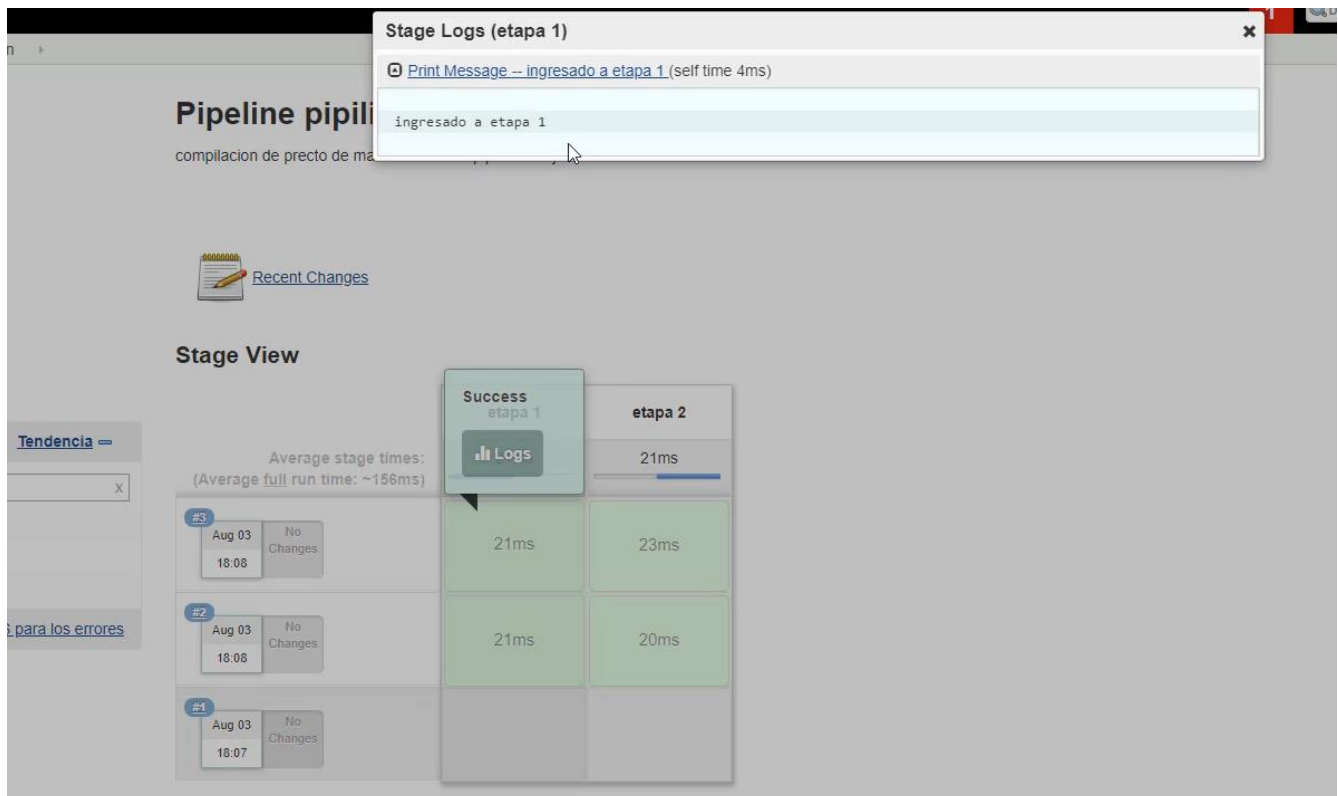
Pipeline script

Script

```
1 node {  
2   echo 'Hello World'  
3   stage('etapa 1'){  
4     echo 'ingresado a etapa 1'  
5   }  
6   stage('etapa 2'){  
7     echo 'ingresando a la etapa 2'  
8   }  
9 }
```

Use Github Sandbox

Al momento de ejecutar nuestro pipeline, nos mostrara de forma grafica, las distintas etapas de nuestra tarea con su respectivo resultado.





Luego modificaremos nuestra tarea para que pueda consumir un proyecto almacenado en GitHub realizando las pruebas correspondientes.

```
1 node {
2   def mvnHome
3   stage('Preparation') { // for display purposes
4     // Get some code from a GitHub repository
5     git 'https://github.com/FranciscoGallardo3180/simple-maven-project-with-tests.git'
6     // Get the Maven tool.
7     // ** NOTE: This 'M3' Maven tool must be configured
8     // **       in the global configuration.
9     mvnHome = tool 'MAVEN'
10  }
11  stage('Build') {
12    // Run the maven build
13    withEnv(["M2_HOME=$mvnHome"]) {
14      if (isUnix()) {
15        sh "$M2_HOME/bin/mvn" -Dmaven.test.failure.ignore clean package
16      } else {
17        bat("/%M2_HOME%\bin\mvn" -Dmaven.test.failure.ignore clean package/)
18      }
19    }
20  }
21  stage('Results') {
22    junit '**/target/surefire-reports/TEST-*.xml'
23    archiveArtifacts 'target/*.jar'
24  }
25 }
```

Nota: en la variable mvnHome = 'MAVEN' hace referencia a nuestro entorno configurado anteriormente en Jenkins, también en el extracto de código M2\_HOME hace referencia a la variable de entorno configurada en nuestro equipo.

```
withEnv(["M2_HOME=$mvnHome"]) {
    if (isUnix()) {
        sh "$M2_HOME/bin/mvn" -Dmaven.test.failure.ignore clean package
    } else {
        bat("/%M2_HOME%\bin\mvn" -Dmaven.test.failure.ignore clean package/)
    }
}
```

Al realizar la ejecución de nuestro pipeline notaremos que este nos entrega un artefacto descargable correspondiente a la compilación del código listo para pasar a producción.

## Pipeline pipeline-compilacion

compilacion de precto de maven desde un pipeline de jenkins.



[Last Successful Artifacts](#)



[simple-maven-project-with-tests-1.0-SNAPSHOT.jar](#)

1,73 KB [view](#)



[Recent Changes](#)



### Stage View



## RESPUESTAS ESPERADAS

1. ¿Cuál es la importancia de crear procesos automatizados para configurar un trabajo en Jenkins que se active automáticamente, en función de ciertas acciones realizadas en el repositorio de SCM, como Bitbucket o GitHub?

Jenkins es capaz de realizar tareas complejas a través de una ejecución simple de código. Además dispone de una gran variedad de plugins que ofrecen la posibilidad de realizar múltiples acciones y una personalización completa del sistema. Finalmente, Jenkins ofrece diversas alternativas para ejecutar sus tareas, desde ejecuciones automatizadas cuando se suban cambios a los repositorios GIT, hasta una API REST para iniciar, actualizar descripciones o habilitar/deshabilitar tareas.

Tener controlados nuestros tests de integración en un entorno de integración continua sin que se conviertan en bloqueantes en caso de que los sistemas de terceros de los que dependen no estén disponibles y provoquen que fallen. Aunque sin perder en ningún momento las alertas ni las métricas de cobertura y éxitos de todos nuestros tests.