

EXERCISES QUE TRABAJAREMOS EN LA CUE

- EXERCISE 1: SUBIENDO EL PROYECTO A GITHUB.
- EXERCISE 2: INSTALANDO JAVA.
- EXERCISE 3: INSTALANDO JENKINS.
- EXERCISE 4: CONFIGURANDO JENKINS.
- EXERCISE 5: CONECTANDO CON GITHUB.

EXERCISE 1: SUBIENDO EL PROYECTO A GITHUB

INTRODUCCIÓN

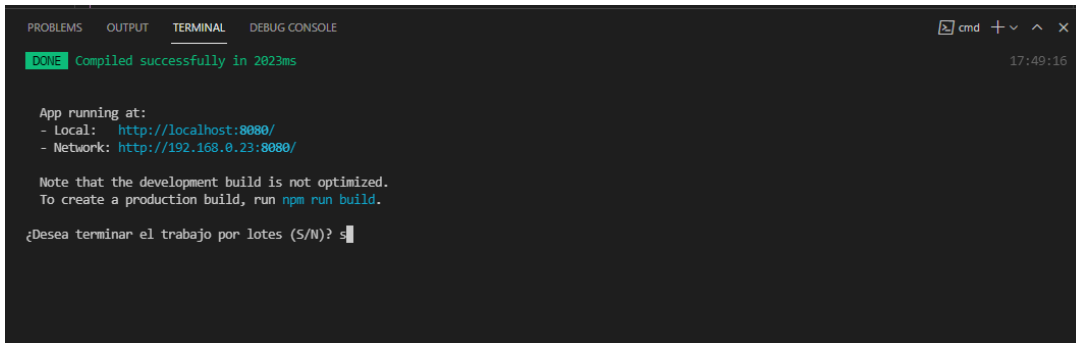
En el siguiente ejercicio, revisaremos cómo incluir una herramienta de integración continua, a un proyecto de **Vue** que tiene las dependencias necesarias para realizar pruebas unitarias. Seguiremos los pasos para instalar, y usar **Jenkins**.

SUBIENDO EL PROYECTO A GITHUB

Para comenzar, reciclaremos la base de nuestro código, de algún ejercicio de un CUE anterior, el cual hemos dejado disponible en este Exercise. Una vez descargado, es necesario descomprimirlo en la carpeta que se quiere destinar para realizar el ejercicio. Con eso realizado, pueden ir a Visual Studio Code, o al editor de texto de su preferencia. Abren la carpeta del ejercicio, la terminal, escriben el comando: **"npm install"** y presionan "enter". Con esto, ya pueden instalar todas las dependencias del proyecto, pues generalmente, en la versión comprimida no incluimos la carpeta **"node modules"**. En este caso, no se procesará nada de eso, pues ya tenemos el proyecto en el computador. Una vez que la instalación de dependencias ha finalizado, podemos levantar el proyecto para confirmar que todo está en orden. Para eso, en la misma terminal, escribimos: **"npm run serve"**, y nuevamente "enter".

Cuando termina de compilar, vamos al puerto en el navegador, y confirmamos que funciona bien.

Presionamos **“control + C”**, y escribimos **“S”** de sí, presionamos “enter”. Con eso, detenemos el modo “serve” y quedamos listos para seguir.



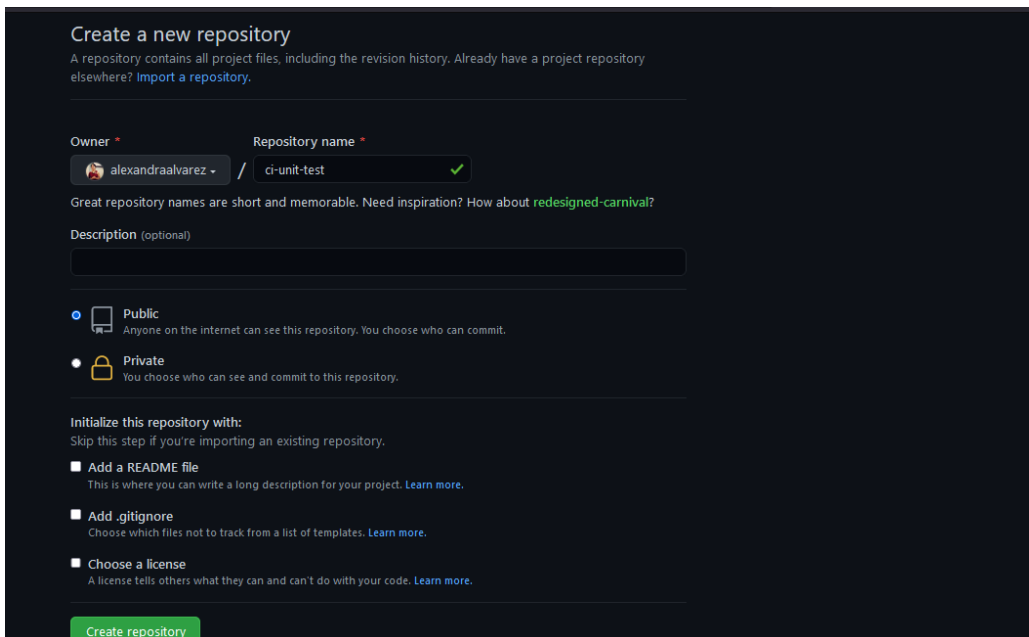
```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
DONE Compiled successfully in 2023ms 17:49:16

App running at:
- Local: http://localhost:8080/
- Network: http://192.168.0.23:8080/

Note that the development build is not optimized.
To create a production build, run npm run build.

¿Desea terminar el trabajo por lotes (S/N)? S
```

Como lo vimos en la lectura, **Jenkins** usa **Git** como controlador de versiones. Por eso, nos dirigimos a nuestra cuenta de **Github**, e iniciamos sesión. Vamos a crear un nuevo repositorio, con el nombre que le queremos dar a nuestro proyecto. En este caso, le pondremos: **“ci-unit-test”**. Los demás parámetros quedan igual, y presionamos crear repositorio.



Create a new repository

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

Owner * alexandraalvarez / Repository name * ci-unit-test ✓

Great repository names are short and memorable. Need inspiration? How about [redesigned-carnival?](#)

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

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

Initialize this repository with:
Skip this step if you're importing an existing repository.

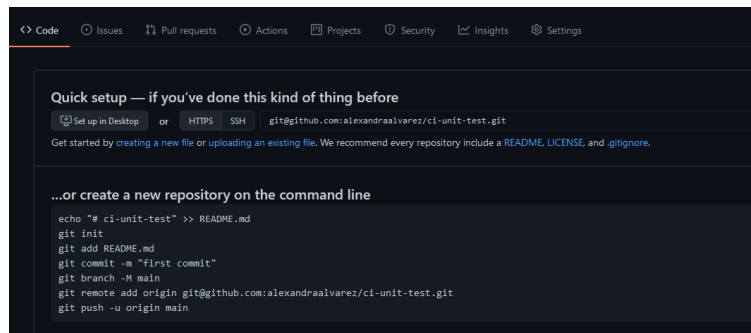
☒ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☒ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

☒ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

De inmediato, aparece una lista de pasos para conectar nuestro repositorio con el proyecto local.
Vamos a seguirlos:

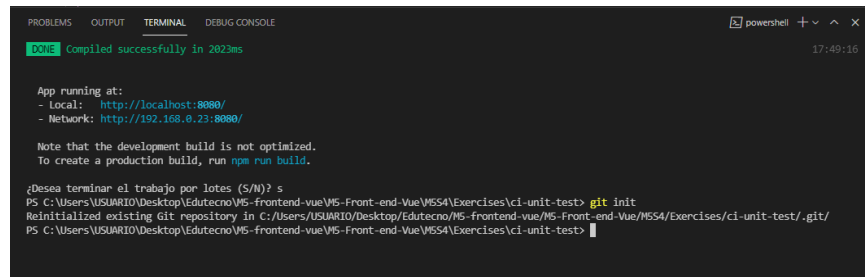


```
<> Code  Issues  Pull requests  Actions  Projects  Security  Insights  Settings

Quick setup — if you've done this kind of thing before
Set up in Desktop  or  HTTPS  SSH  git@github.com:alexandraalvarez/ci-unit-test.git
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line
echo "# ci-unit-test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:alexandraalvarez/ci-unit-test.git
git push -u origin main
```

Copiamos **“git init”**, lo pegamos en la terminal, y presionamos “enter”.



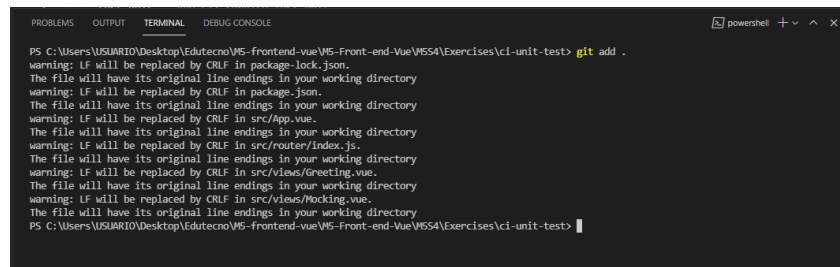
```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  powershell + v ^ x
DONE Compiled successfully in 2023ms 17:49:16

App running at:
- Local: http://localhost:8080/
- Network: http://192.168.0.23:8080/

Note that the development build is not optimized.
To create a production build, run npm run build.

¿Desea terminar el trabajo por lotes (S/N)? s
PS C:\Users\USUARIO\Desktop\Edutecno\VS-front-end-vue\VS-Front-end-Vue\VS54\Exercises\ci-unit-test> git init
Reinitialized existing Git repository in C:\Users\USUARIO\Desktop\Edutecno\VS-front-end-vue\VS-Front-end-Vue\VS54\Exercises\ci-unit-test\.git/
PS C:\Users\USUARIO\Desktop\Edutecno\VS-front-end-vue\VS-Front-end-Vue\VS54\Exercises\ci-unit-test> 
```

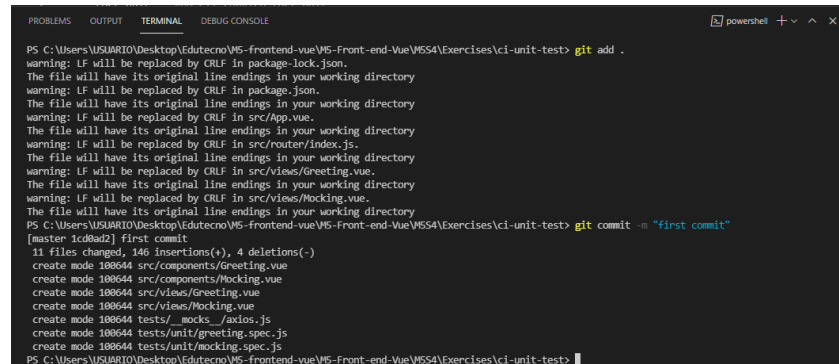
Continuamos escribiendo el comando **“git add .”**, y presionamos “enter”.



```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  powershell + v ^ x

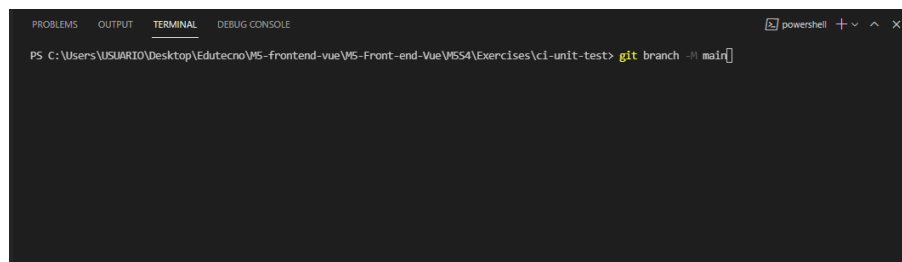
PS C:\Users\USUARIO\Desktop\Edutecno\VS-front-end-vue\VS-Front-end-Vue\VS54\Exercises\ci-unit-test> git add .
warning: LF will be replaced by CRLF in package-lock.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in package.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/app.vue.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/router/index.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/views/Greeting.vue.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/views/Loading.vue.
The file will have its original line endings in your working directory
PS C:\Users\USUARIO\Desktop\Edutecno\VS-front-end-vue\VS-Front-end-Vue\VS54\Exercises\ci-unit-test> 
```

De inmediato copiamos `“git commit -m “first commit””`, lo pegamos en la terminal de Visual Studio Code, y presionamos “enter”.



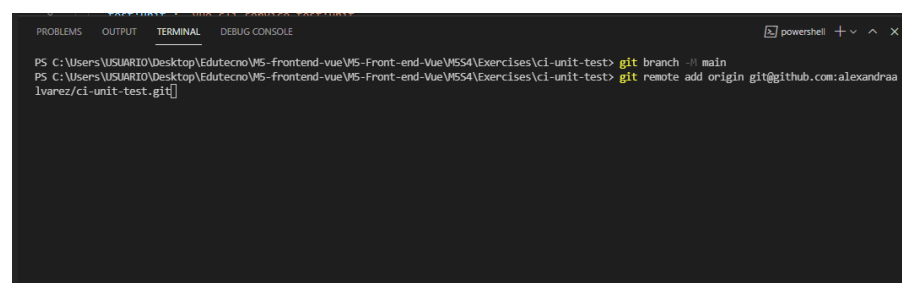
```
PS C:\Users\USUARIO\Desktop\Edutecno\VS-front-end-vue\VS-Front-end-Vue\VS4\Exercises\ci-unit-test> git add .
warning: LF will be replaced by CRLF in package-lock.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in package.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/App.vue.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/router/index.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/views/Greeting.vue.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/views/Mocking.vue.
The file will have its original line endings in your working directory
PS C:\Users\USUARIO\Desktop\Edutecno\VS-front-end-vue\VS-Front-end-Vue\VS4\Exercises\ci-unit-test> git commit -m "first commit"
[master 1c08ad2] first commit
11 files changed, 146 insertions(+), 4 deletions(-)
create mode 100644 src/components/Greeting.vue
create mode 100644 src/components/Mocking.vue
create mode 100644 src/views/Greeting.vue
create mode 100644 src/views/Mocking.vue
create mode 100644 tests/_mocks_/axios.js
create mode 100644 tests/unit/greeting.spec.js
create mode 100644 tests/unit/mocking.spec.js
PS C:\Users\USUARIO\Desktop\Edutecno\VS-front-end-vue\VS-Front-end-Vue\VS4\Exercises\ci-unit-test>
```

Copiamos `“git branch -M main”`, lo pegamos en la terminal, y presionamos “enter”.



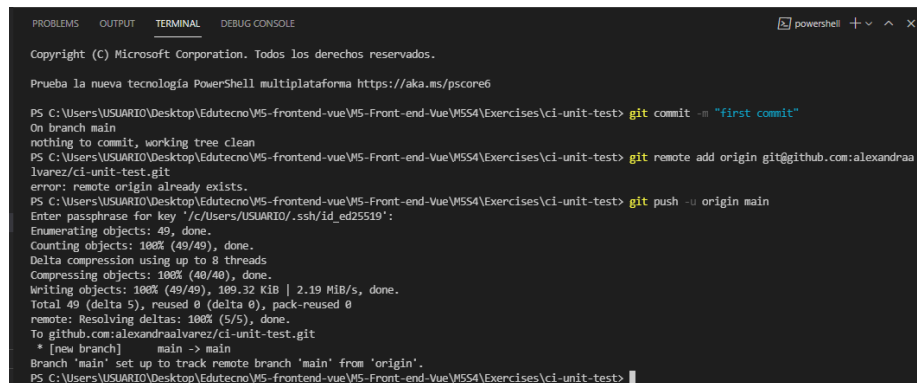
```
PS C:\Users\USUARIO\Desktop\Edutecno\VS-front-end-vue\VS-Front-end-Vue\VS4\Exercises\ci-unit-test> git branch -M main
```

El comando que sigue es: `“git remote add origin git@github.com:alexandraalvarez/ci-unit-test.git”`. Procedemos igual que con los comandos anteriores, y presionamos “enter”.



```
PS C:\Users\USUARIO\Desktop\Edutecno\VS-front-end-vue\VS-Front-end-Vue\VS4\Exercises\ci-unit-test> git branch -M main
PS C:\Users\USUARIO\Desktop\Edutecno\VS-front-end-vue\VS-Front-end-Vue\VS4\Exercises\ci-unit-test> git remote add origin git@github.com:alexandraalvarez/ci-unit-test.git
```

Copiamos la última línea de comando: `git push -u origin main`, la pegamos en la terminal, y presionamos “enter”.

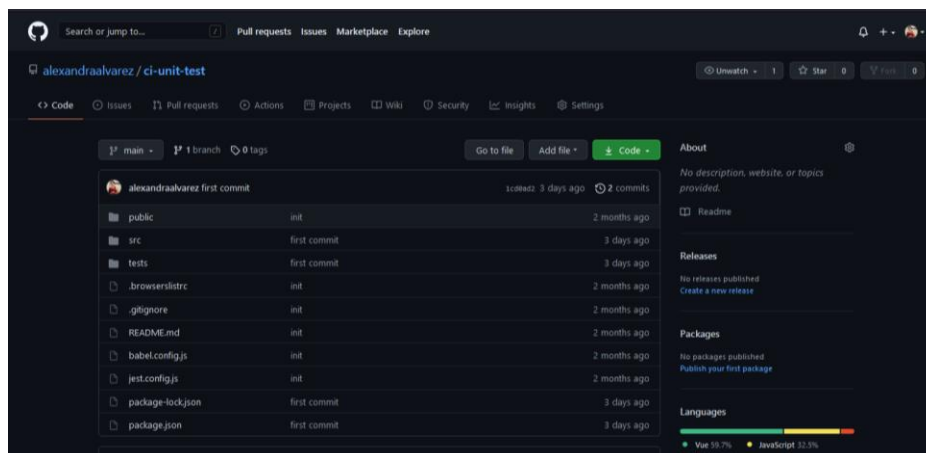


```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE powershell + - ^ x
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\USUARIO\Desktop\EducTecno\VS-front-end-vue\VS4\Exercises\ci-unit-test> git commit -m "first commit"
On branch main
nothing to commit, working tree clean
PS C:\Users\USUARIO\Desktop\EducTecno\VS-front-end-vue\VS4\Exercises\ci-unit-test> git remote add origin git@github.com:alexandraa
lvarez/ci-unit-test.git
error: remote origin already exists.
PS C:\Users\USUARIO\Desktop\EducTecno\VS-front-end-vue\VS4\Exercises\ci-unit-test> git push -u origin main
Enter passphrase for key '/c/Users/USUARIO/.ssh/id_ed25519':
Enumerating objects: 49, done.
Counting objects: 100% (49/49), done.
Delta compression using up to 8 threads
Compressing objects: 100% (40/40), done.
Writing objects: 100% (49/49), 109.32 KiB | 2.19 MiB/s, done.
Total 49 (delta 5), reused 8 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), done.
To github.com:alexandraalvarez/ci-unit-test.git
 * [new branch]    main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
PS C:\Users\USUARIO\Desktop\EducTecno\VS-front-end-vue\VS4\Exercises\ci-unit-test> |
```

Como se puede notar, en este caso pidió ingresar la clave `ssh`. Una vez escrita, vemos que el contenido local de nuestro proyecto sube al repositorio. Si lo vamos a revisar, nos daremos cuenta de que está en el repositorio en `Github`, actualizado hace días, pero es porque la carpeta estaba creada previamente, y con un `push` sin terminar.



De inmediato, vamos a instalar `Java`.

EXERCISE 2: INSTALANDO JAVA

INSTALANDO JAVA

El siguiente paso para poder instalar **Jenkins**, es instalar **Java SE Development Kit**. Nos dirigimos a la página: <https://www.oracle.com/cl/java/technologies/javase-downloads.html>. Buscamos la versión **LTS**, que corresponde a la más estable de las liberadas, en este caso es la 11, y le damos clic al enlace: "JDK download".

Java SE 11 (LTS)

Java SE 11.0.11 is the latest release for the Java SE 11 Platform

- [Documentation](#)
- [Installation Instructions](#)
- [Release Notes](#)
- [Oracle License](#)
 - [Binary License](#)
 - [Documentation License](#)
- [Java SE Licensing Information User Manual](#)
 - [Includes Third Party Licenses](#)
- [Certified System Configurations](#)
- [Readme](#)

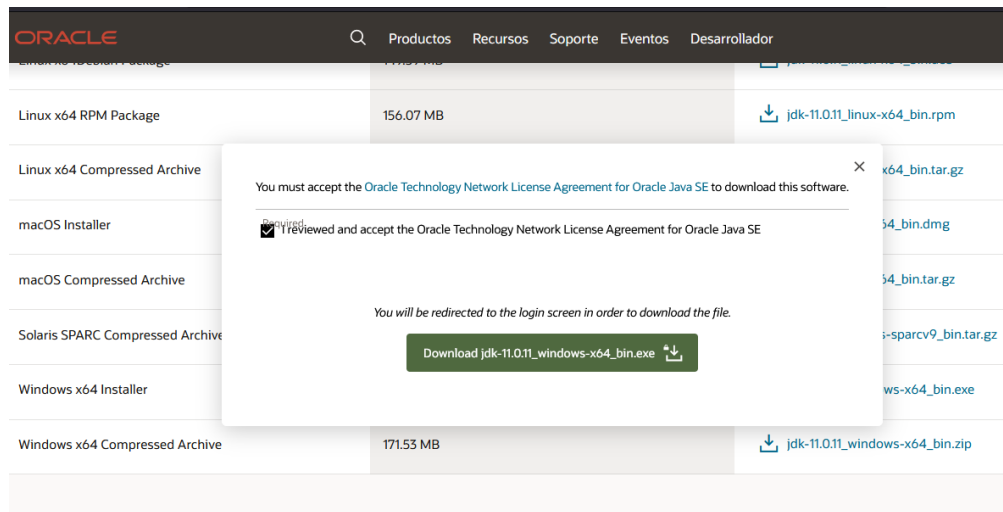
Oracle JDK

- [JDK Download](#)
- [Documentation Download](#)

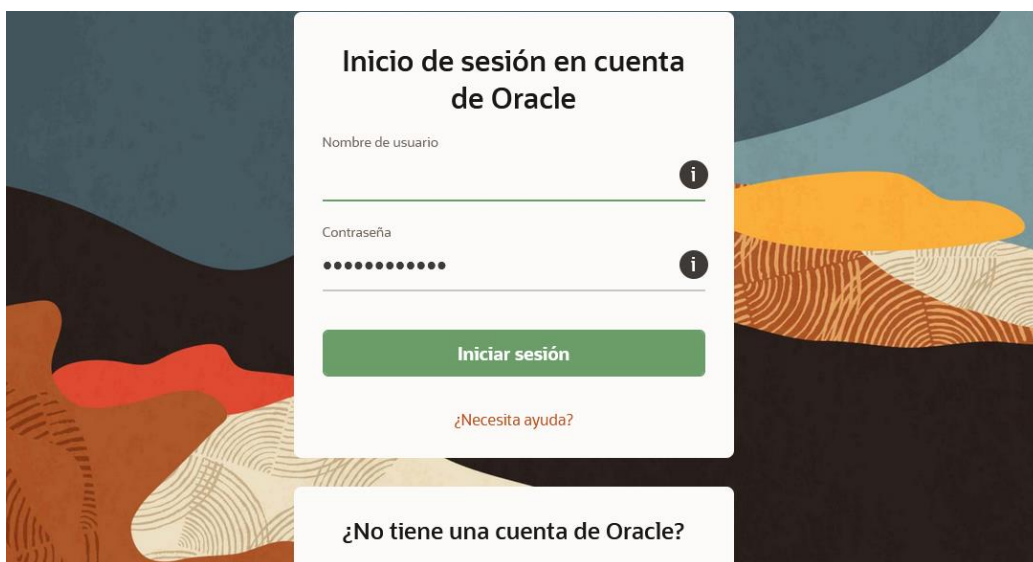
De inmediato, bajamos en la lista de versiones hasta encontrar la de Windows. Pinchamos el enlace.

ORACLE			Productos	Recursos	Soporte	Eventos	Desarrollador
Linux x64 RPM Package			156.07 MB				jdk-11.0.11_linux-x64_bin.rpm
Linux x64 Compressed Archive			173.46 MB				jdk-11.0.11_linux-x64_bin.tar.gz
macOS Installer			167.15 MB				jdk-11.0.11_osx-x64_bin.dmg
macOS Compressed Archive			167.67 MB				jdk-11.0.11_osx-x64_bin.tar.gz
Solaris SPARC Compressed Archive			184.51 MB				jdk-11.0.11_solaris-sparcv9_bin.tar.gz
Windows x64 Installer			152.05 MB				jdk-11.0.11_windows-x64_bin.exe
Windows x64 Compressed Archive			171.53 MB				jdk-11.0.11_windows-x64_bin.zip

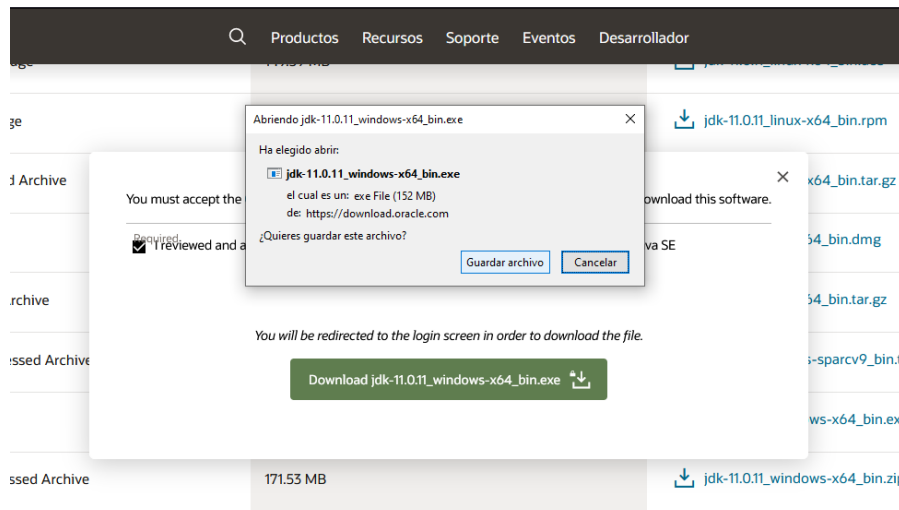
Se despliega una nueva ventana. Aceptamos las condiciones de licencia de Oracle, y marcamos el botón: **“Download JDK”**.



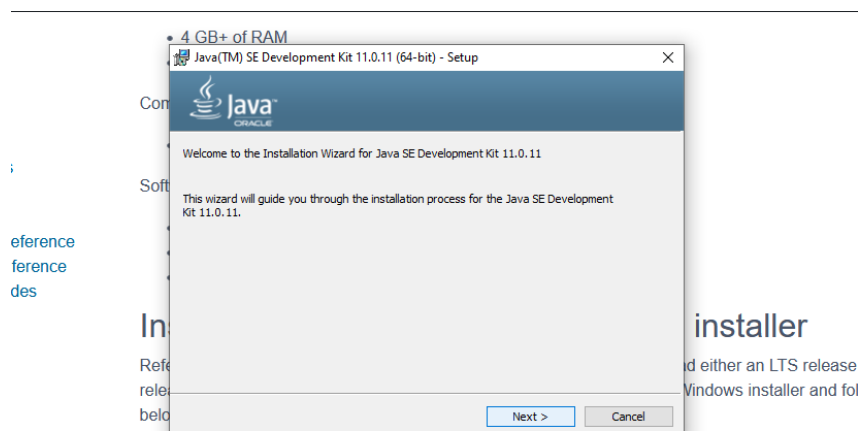
Para continuar con la descarga, es necesario iniciar sesión en Oracle. Para ello, deben crear una cuenta personal y conectarse.



Si iniciamos sesión de manera correcta, nos dará inmediatamente la opción de comenzar la descarga.



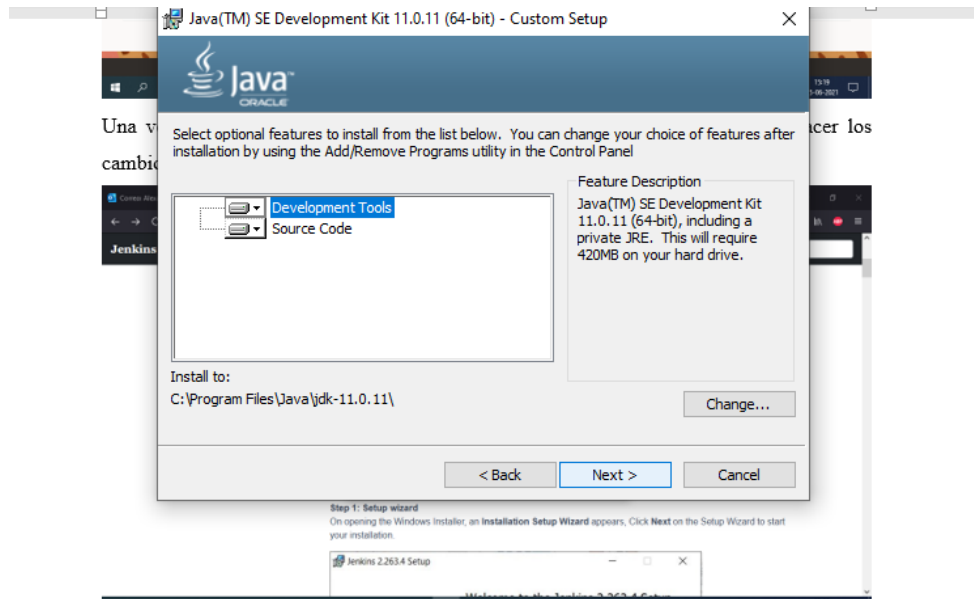
Una vez descargado el archivo, lo ejecutamos, y le damos los permisos para hacer los cambios que necesita. En la primera vista presionamos “Next”.



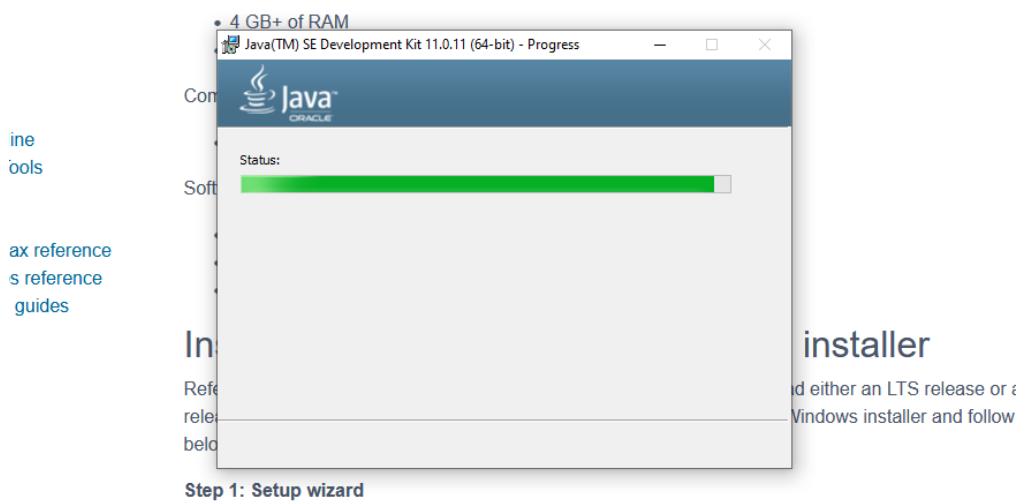
Step 1: Setup wizard

On opening the Windows Installer, an **Installation Setup Wizard** appears, Click **Next** on the Setup:

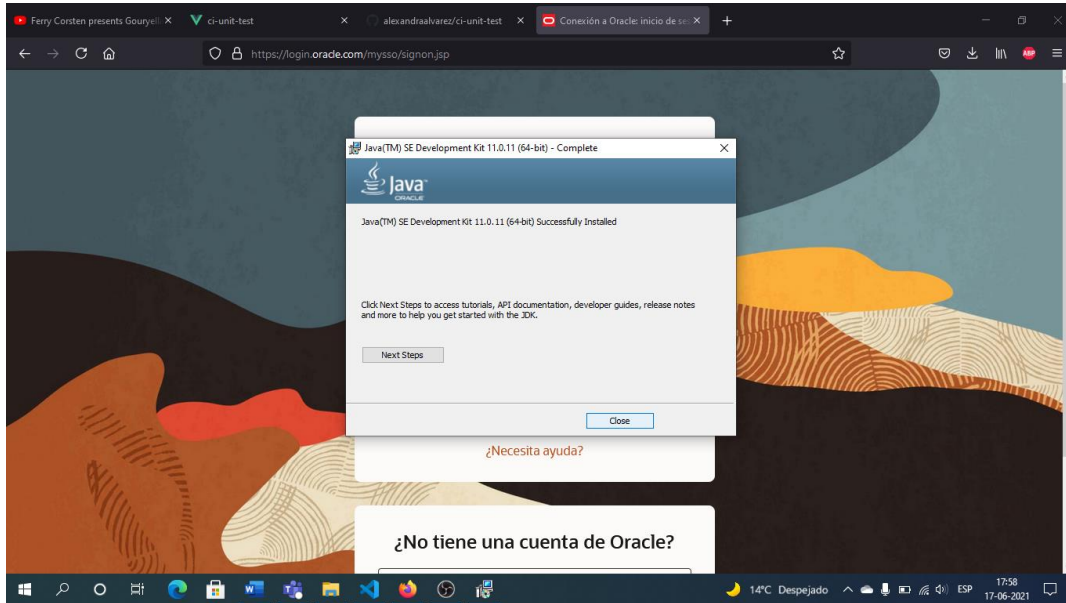
En la siguiente, dejamos la ruta que nos provee por defecto, y presionamos "Next".



Vemos como se comienza a instalar inmediatamente.

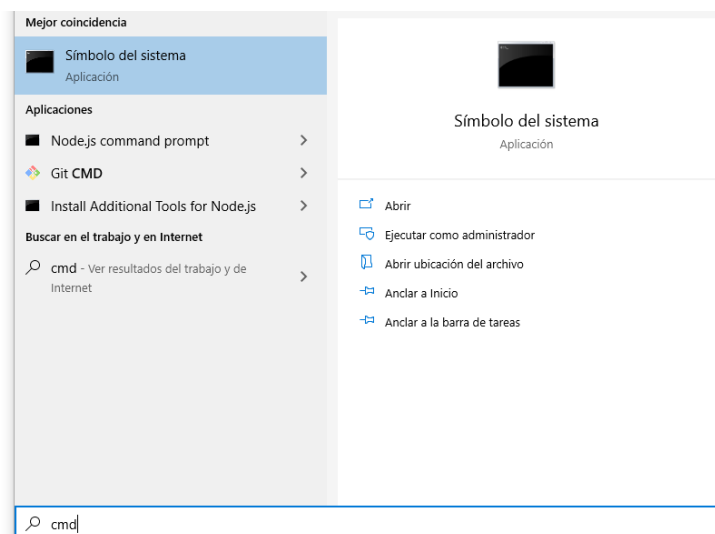


Cuando finaliza la instalación, presionamos “Close”.

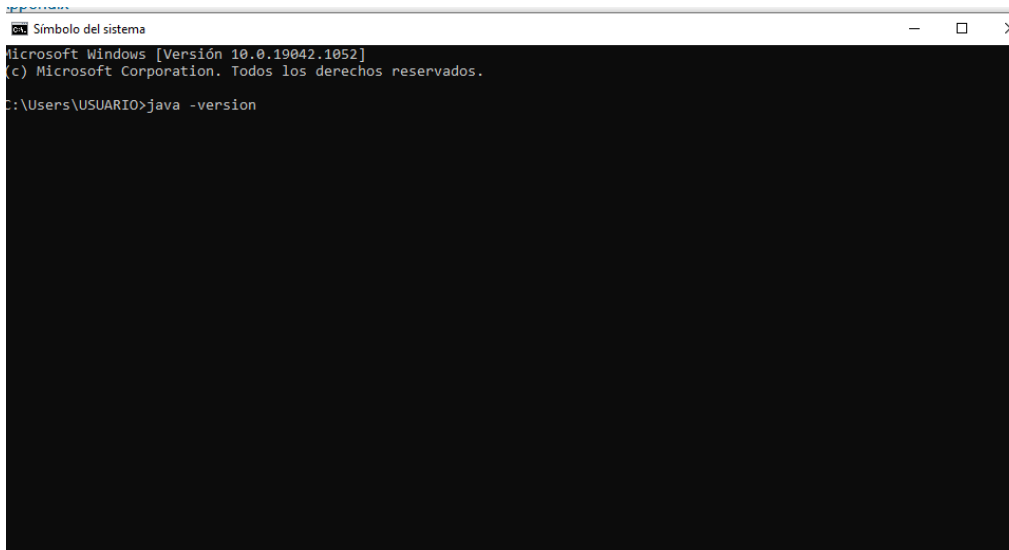


CONFIRMANDO INSTALACIÓN

De inmediato, abrimos la consola, tecleando **“cmd”** en el buscador de la barra de programas de Windows.



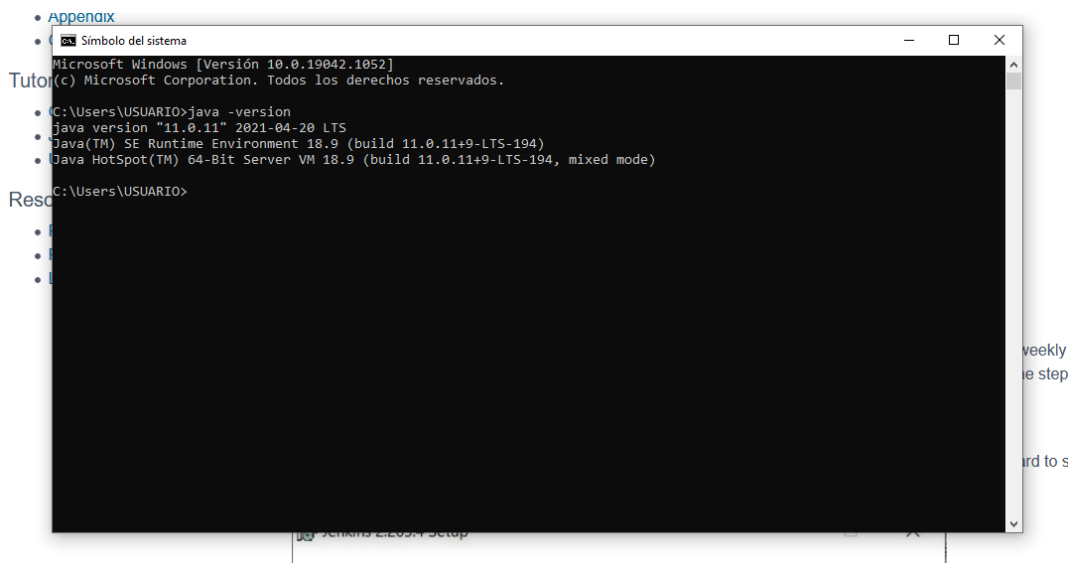
Para comprobar que la instalación de **JDK** está correcta, vamos a escribir: **"java -version"**.



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19042.1052]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\USUARIO>java -version
```

Y presionamos "enter", obteniendo como respuesta la versión del programa instalado.



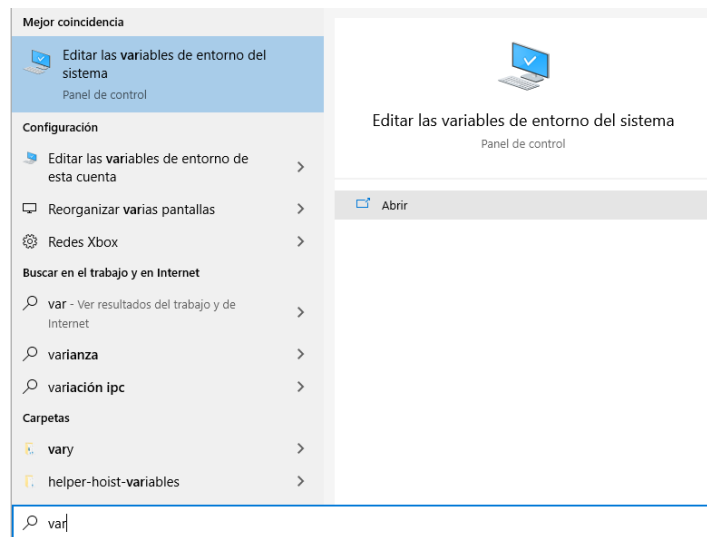
```

• Appendix
• Símbolo del sistema
Tutoriales
Microsoft Windows [Versión 10.0.19042.1052]
(c) Microsoft Corporation. Todos los derechos reservados.

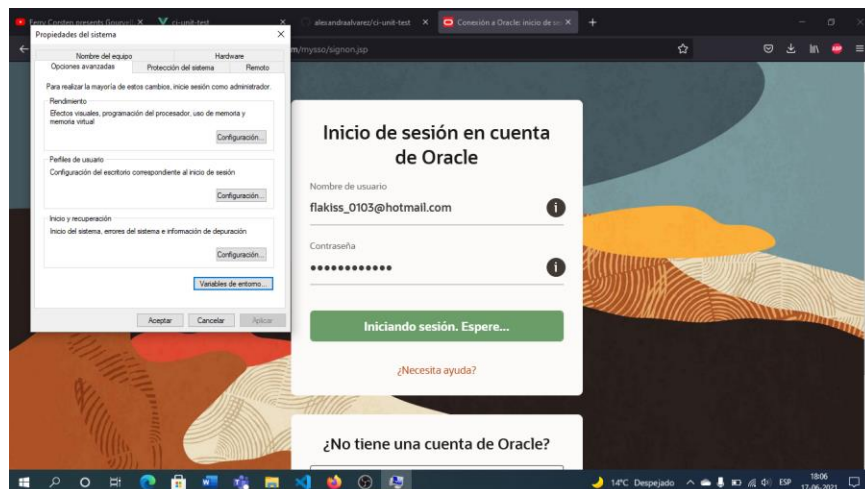
C:\Users\USUARIO>java -version
java version "11.0.11" 2021-04-20 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.11+9-LTS-194)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.11+9-LTS-194, mixed mode)
C:\Users\USUARIO>
Res
•
•
•
•
weekly
the step
ard to s
```

CONFIGURANDO VARIABLES DE AMBIENTE

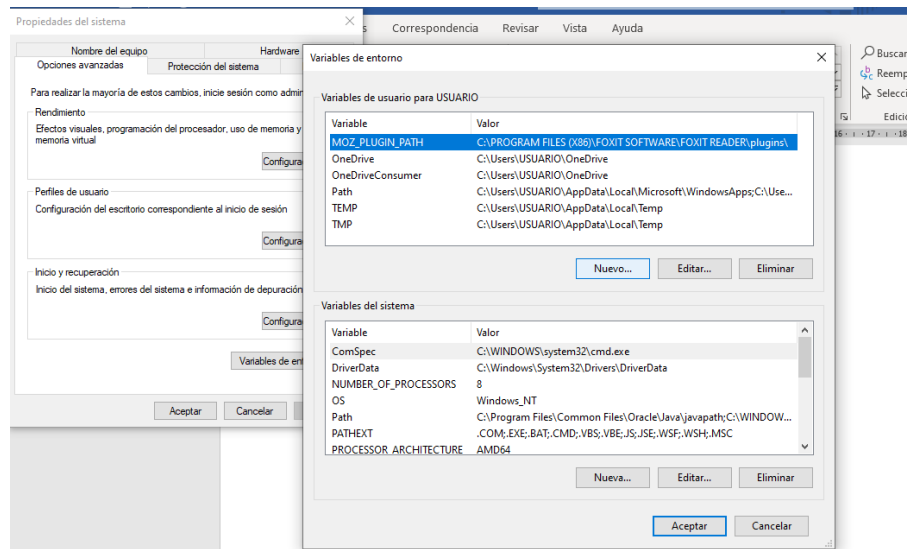
En algunos casos, será necesario configurar las variables de ambiente, así que lo revisaremos en este momento. Para eso, vamos al buscador de programas, escribimos **“variables”**, y presionamos abrir.



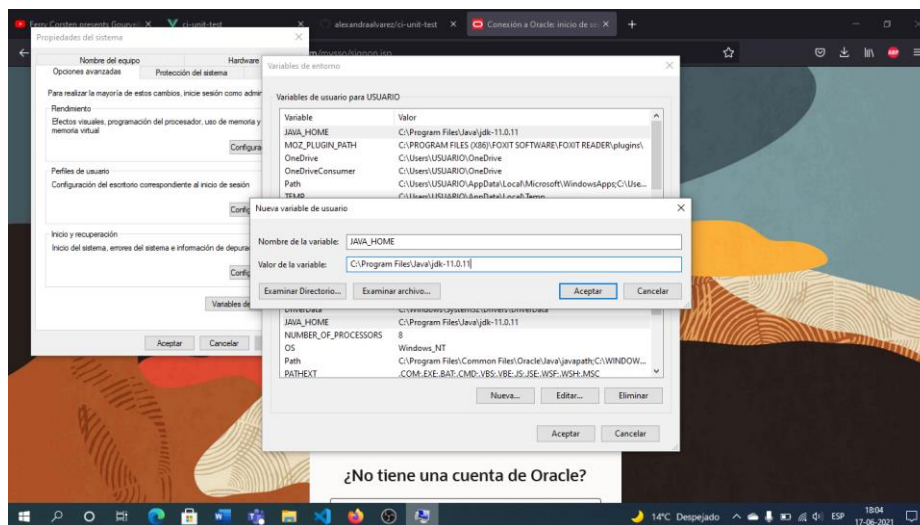
En la ventana que se abre, vamos al botón **“variables de entorno”**, y le hacemos clic.



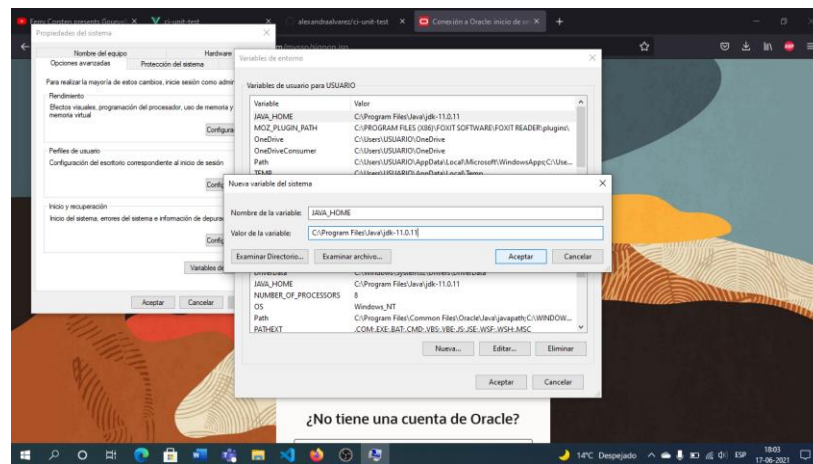
Se abre una nueva ventana, donde agregaremos una variable de usuario, para eso hacemos clic sobre el botón “Nuevo...”.



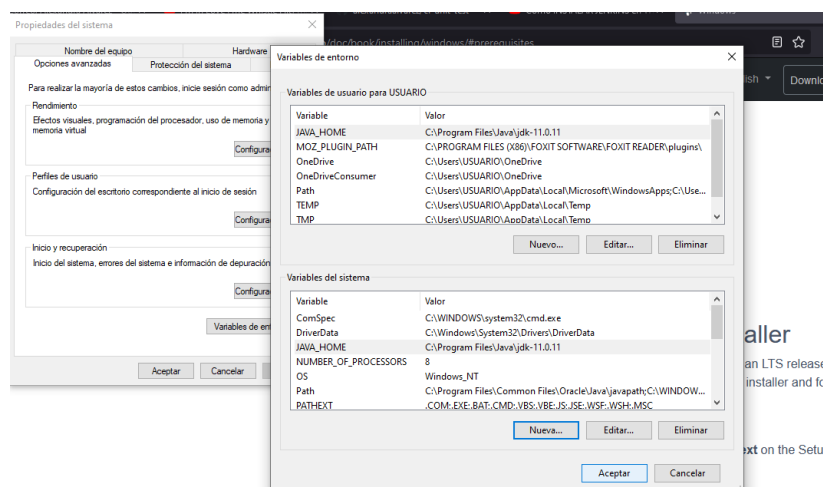
Escribimos **“JAVA_HOME”**, en el nombre de la variable de usuario y en el valor de la variable, iremos a la capeta de almacenamiento de **Java**, y copiaremos su ruta. En este caso, la ruta es: **“C:>Program Files>Java>jdk-11.0.11”**. Ahora presionamos “Aceptar”, y continuamos.



Hacemos lo mismo con la variable del sistema. Presionamos el botón “Nueva...”, en el nombre de la variable de sistema **“JAVA_HOME”** y en el valor de la variable, iremos a la carpeta de almacenamiento de Java, y copiaremos su ruta. En este caso, la ruta es: **“C:>Program Files>Java>jdk-11.0.11”**. Ahora presionamos “Aceptar”, y continuamos.



De inmediato cerramos las ventanas originales, presionando “Aceptar” en cada una.



Y con eso, ya hemos terminado la configuración de Java.

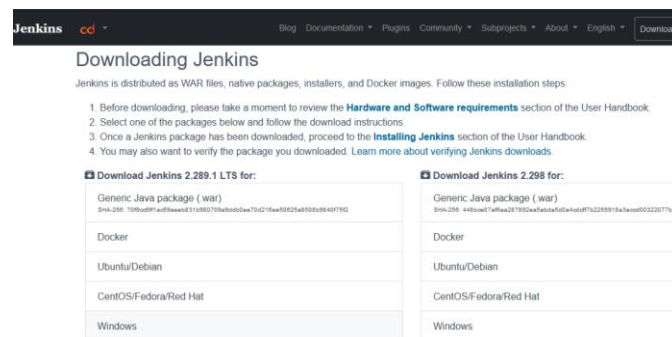
EXERCISE 3: INSTALANDO JENKINS

INTRODUCCIÓN

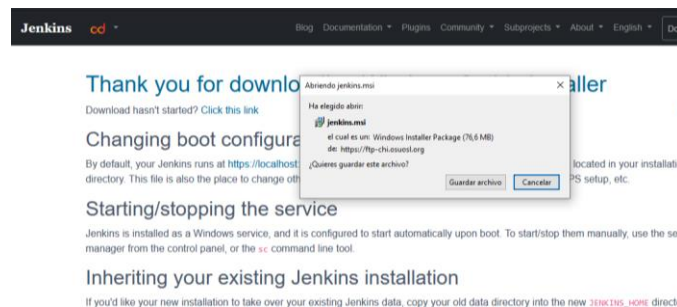
Con el repositorio listo, y **Java** instalado, estamos listos para instalar **Jenkins**.

DESCARGANDO JENKINS

En primer lugar, iremos a la documentación de **Jenkins**, y seguiremos las instrucciones para descargar la versión **LTS**. Ésta se libera anualmente, y es la forma más estable posible, por eso es la que instalaremos. Seleccionamos la que corresponde a Windows, que es el sistema operativo de este computador. En el caso de que tengan otro sistema operativo, también encontrarán opciones compatibles.



Pinchamos sobre “Windows”, y vemos que aparece una ventana que nos pide permiso para guardar el archivo. Se lo concedemos, presionando “Guardar archivo”.



El sistema nos pregunta si confiamos en el archivo, como es de una fuente oficial, le damos clic al botón “Aceptar y seguimos”.

Thank you for downloading Windows Stable installer

Download hasn't started? [Click this link](#)

[Tweet](#)

Changing boot configuration

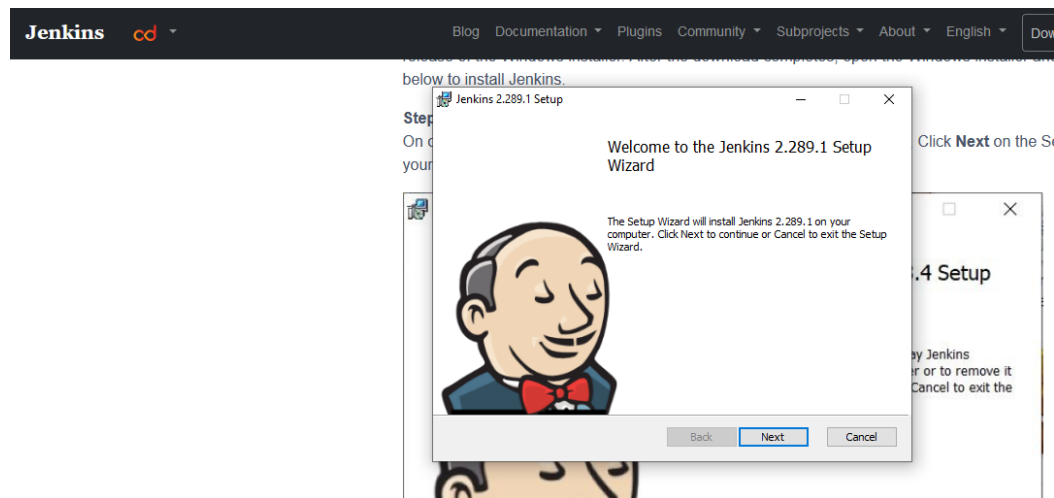
By default, your Jenkins runs at [https://jenkins.example.com](#). This file is also the place to

Starting/stopping the Jenkins service

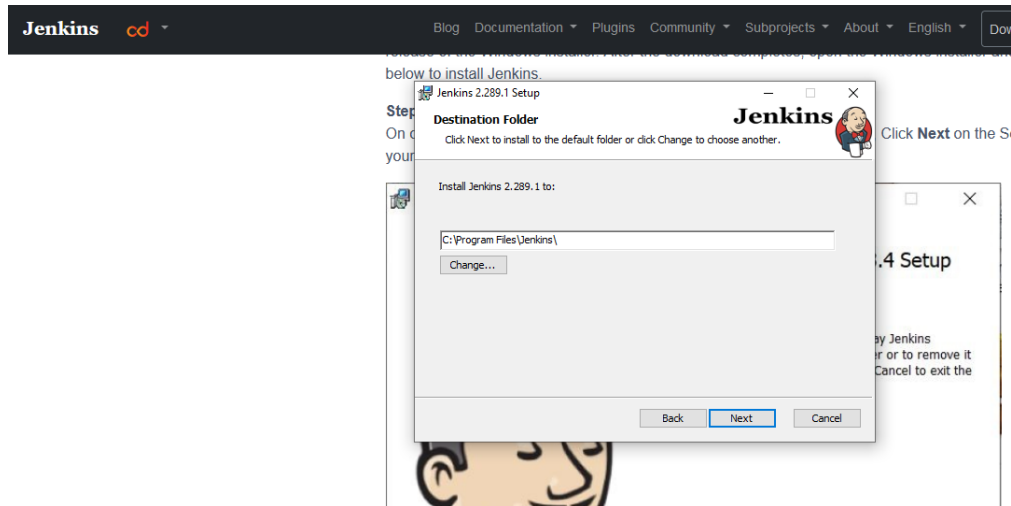
Jenkins is installed as a Windows service, and it is configured to start automatically upon boot. To start/stop them manually, use the service manager from the control panel, or the `sc` command line tool.

inheriting your existing Jenkins installation

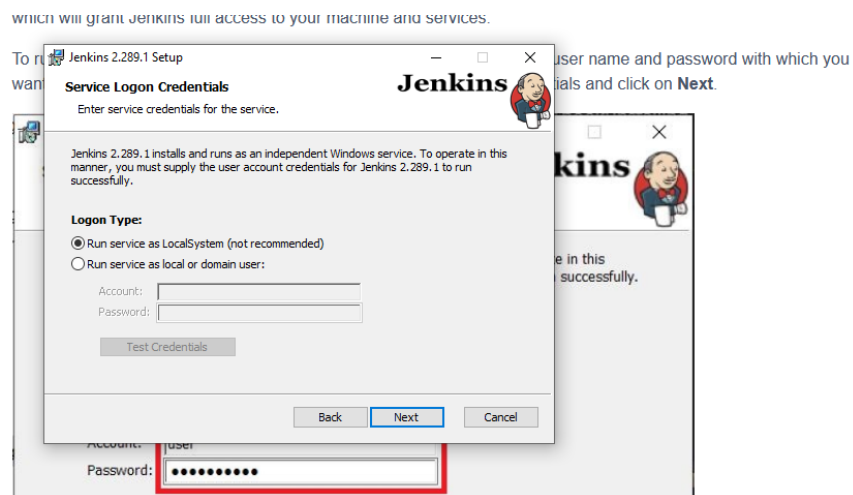
De inmediato, vamos a ejecutar el archivo, y seguir los pasos de instalación. Comenzando por aceptar la ayuda del **Wizard**, cliqueando sobre el botón “next”.



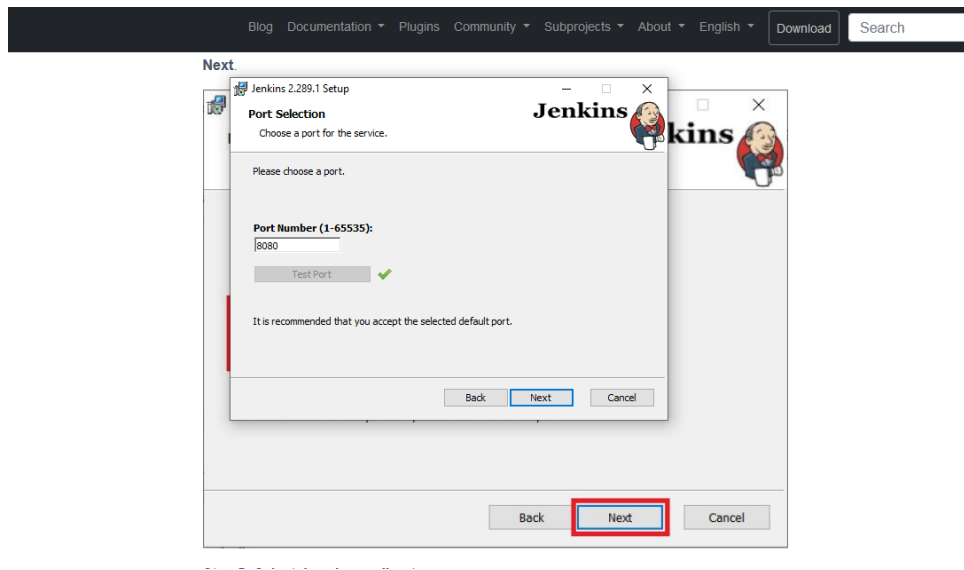
En la siguiente vista, aceptamos la carpeta de destino presionando “next”.



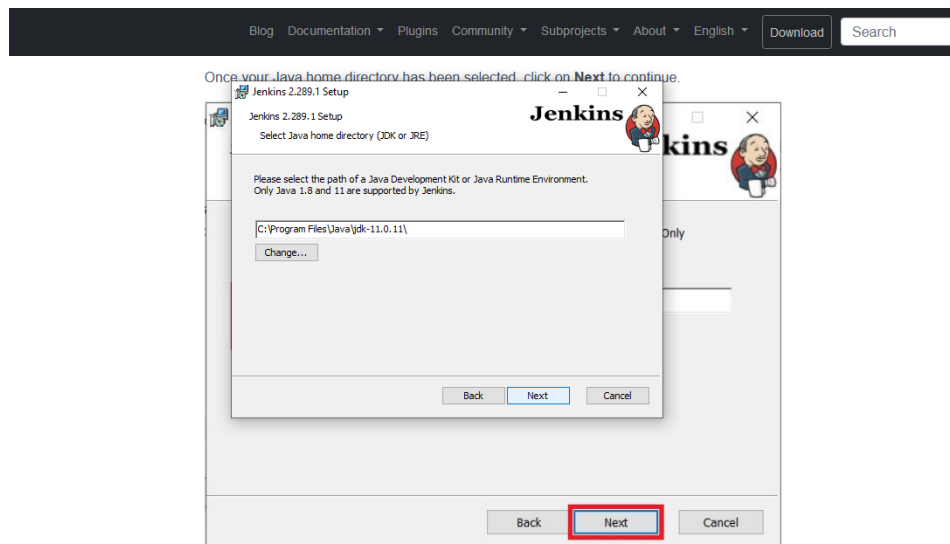
Se recomienda instalar y ejecutar **Jenkins**, como un servicio de Windows independiente, usando un usuario local o de dominio, ya que es mucho más seguro que hacerlo usando **LocalSystem** (equivalente de Windows a root). Sin embargo, como este ejercicio es solo con propósitos pedagógicos, lo instalaremos de esta forma. Así seleccionamos: **“Run service as local system”**, y presionamos “Next”.



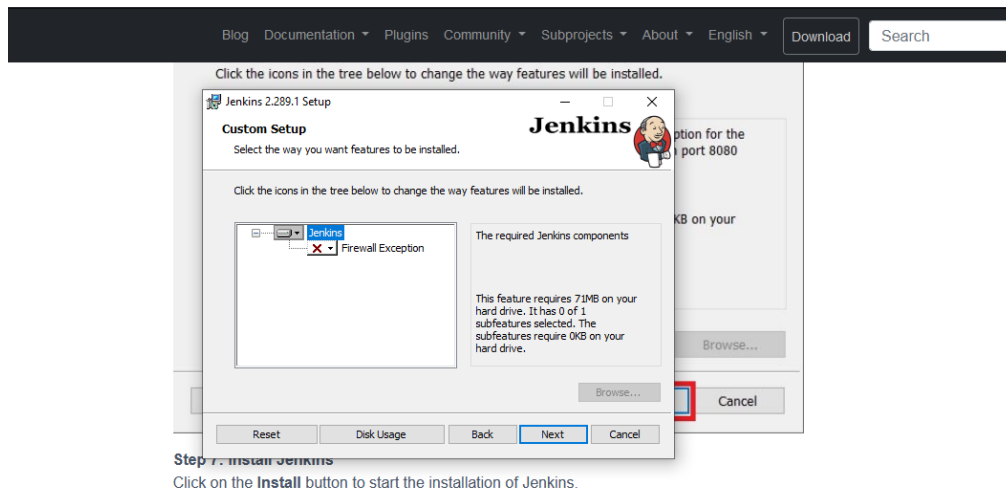
De inmediato, verificamos que el puerto esté disponible, presionando: **“Test Port”**. Con el ticket verde, podemos continuar presionando el botón **“Next”**.



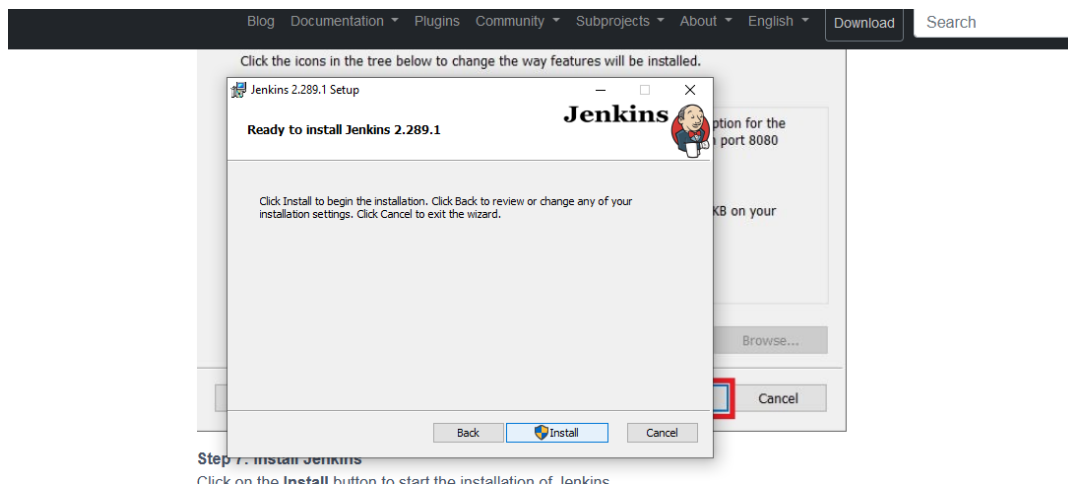
En la siguiente vista necesitamos confirmar la ruta de instalación. Como nosotros ya configuramos **JDK**, nos sugiere la misma ruta, así que presionamos **“Next”**.



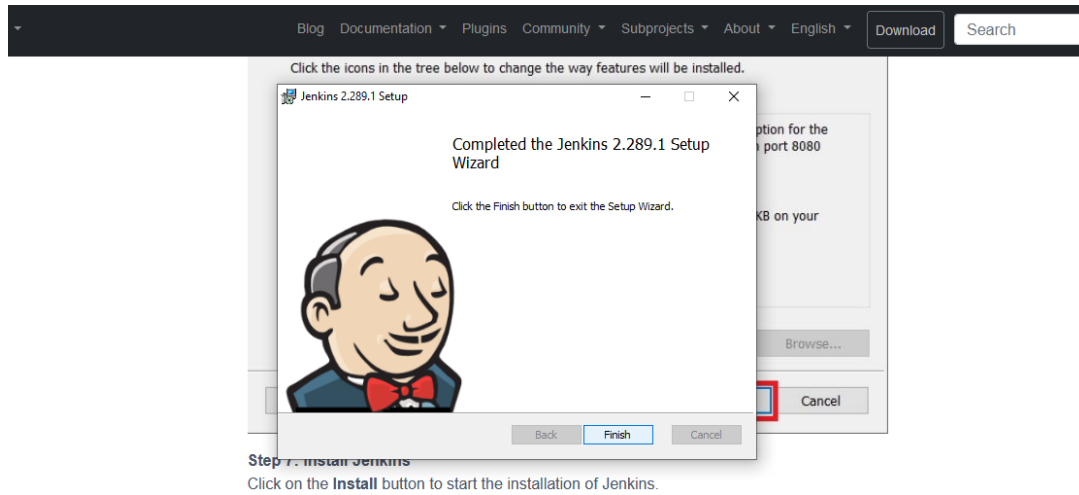
Confirmamos la forma de instalación sugerida presionando “Next”.



Y luego, **“Install”**.



Cuando la instalación termina, presionamos "Finish".

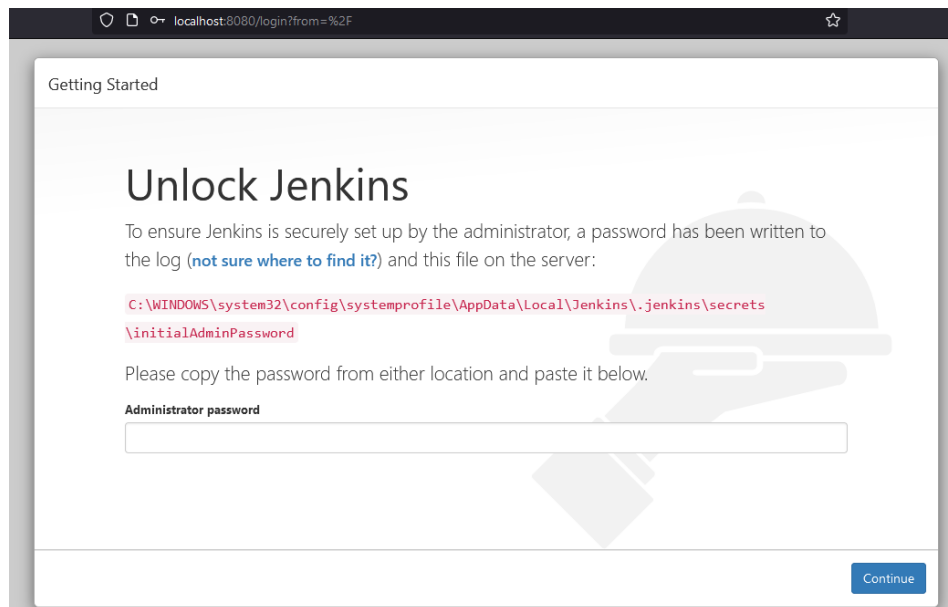


Nuevamente, nos pide autorización para realizar cambios en el sistema. Lo autorizamos, y rápidamente lo instala.

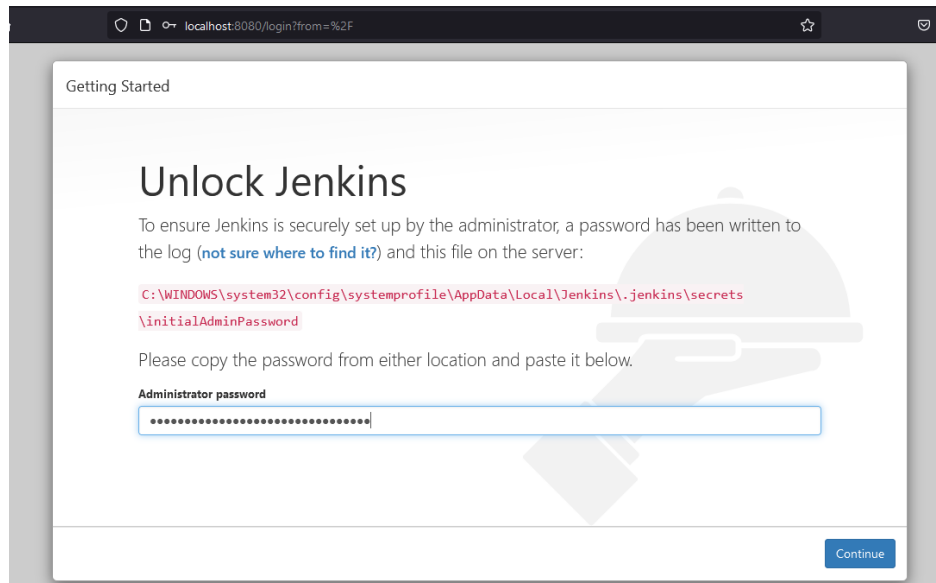
EXERCISE 4: CONFIGURANDO JENKINS

INTRODUCCIÓN

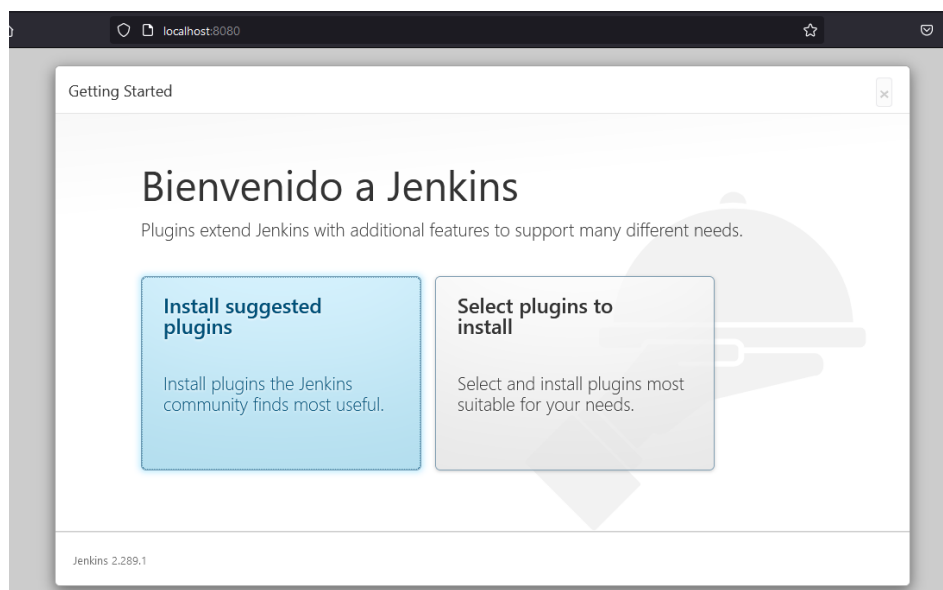
Ahora que hemos logrado instalar **Jenkins**, vamos a desbloquearlo y configurarlo. Como vimos al finalizar la instalación, se abre automáticamente el puerto que, en este caso, es el: `"http://localhost:8080/login?from=%2F"`.



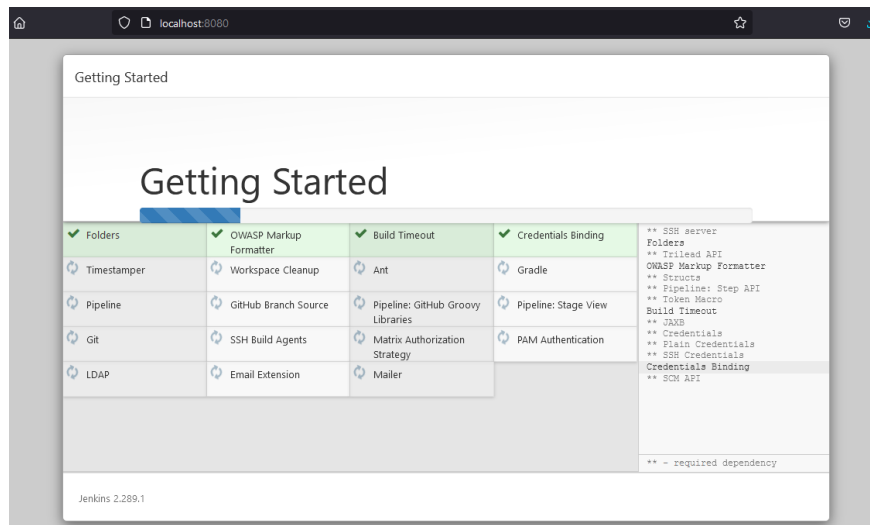
Notaremos que dentro del mensaje de la ventana, nos entrega una ruta en nuestro equipo, donde se oculta una clave que corresponde al password inicial del administrador. Así que, para conseguirlo, seguiremos el camino accediendo a las distintas carpetas mencionadas. En el acceso a las carpetas de `"config"` y `"systemprofile"`, nos pedirá, la primera vez, dar acceso. Al presionar "Continue", accedemos, y desde ahí no volverá a aparecer ese mensaje. Una vez que encontremos el documento, lo abrimos con el block de notas. Copiamos la clave, y la pegamos en el campo donde lo solicitan. Finalizamos pinchando el botón "Continue".



De inmediato notaremos que nos envía a una nueva vista, donde nos pregunta por los **plugins** que queremos instalar, ofreciendo los sugeridos, o la posibilidad de elegir nosotros. Vamos a instalar los **plugins** recomendados.



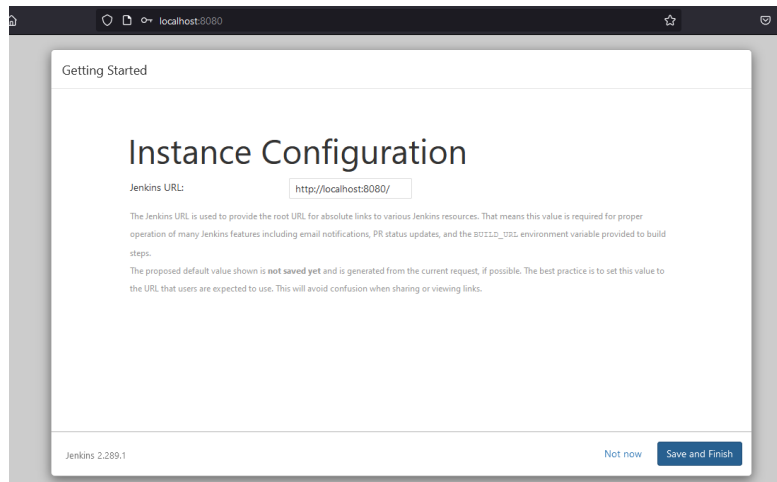
Podemos notar que aparece una nueva vista, donde se visualizan los **plugins** que se están instalando, y el progreso global del proceso.



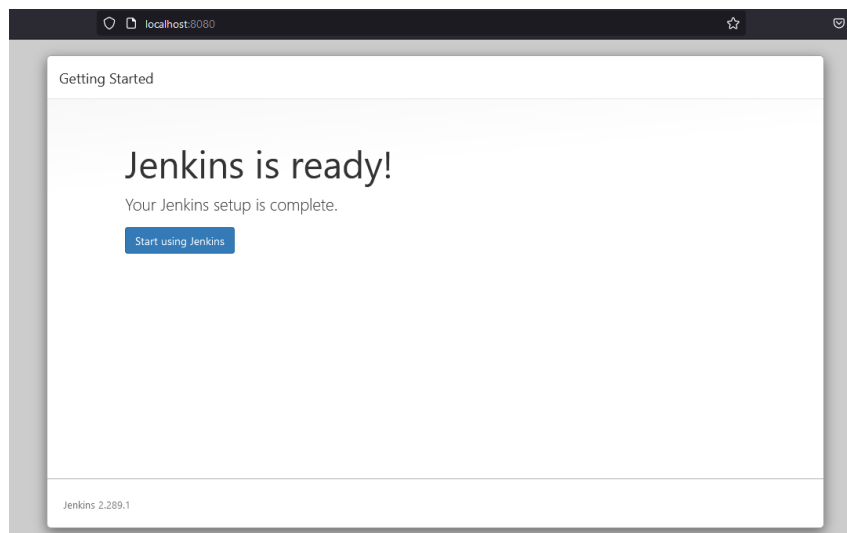
La siguiente etapa, corresponde a crear el usuario y contraseña del administrador, que definirán ustedes mismos. Además, deben ingresar una dirección de correo, y su nombre; es este caso, se usarán las credenciales que tenemos en Oracle, así será más fácil recordarlos.

The screenshot shows the Jenkins 'Create First Admin User' form. The form has five input fields: 'Usuario:', 'Contraseña:', 'Confirma la contraseña:', 'Nombre completo:', and 'Dirección de email:'. At the bottom right, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'. The Jenkins version 'Jenkins 2.289.1' is displayed at the bottom left.

Luego, nos lleva a otra vista, donde podemos revisar la **URL** que usará **Jenkins**. Por defecto, nos sugiere el puerto: <http://localhost:8080/>. Aceptaremos la sugerencia, presionando el botón **“Save and Finish”**.



El programa nos envía a una página, donde nos confirma que estamos listos para comenzar a usar **Jenkins**.

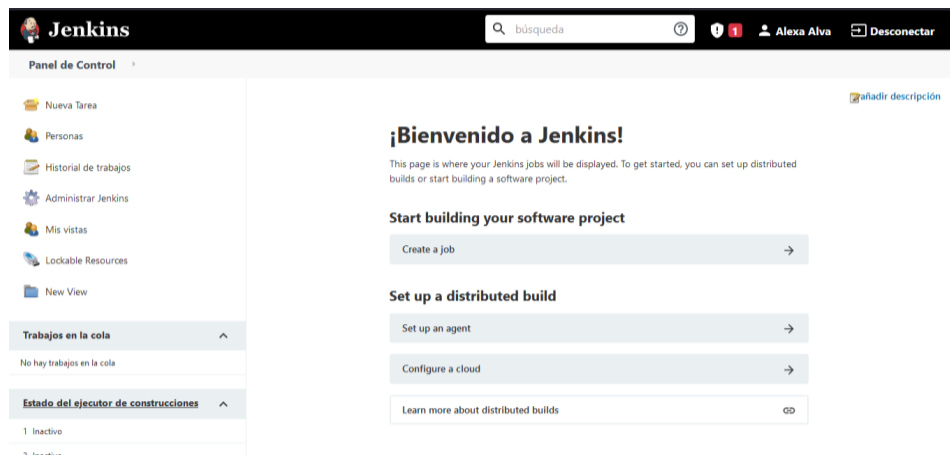


Así terminamos la configuración.

EXERCISE 5: CONECTANDO CON GITHUB

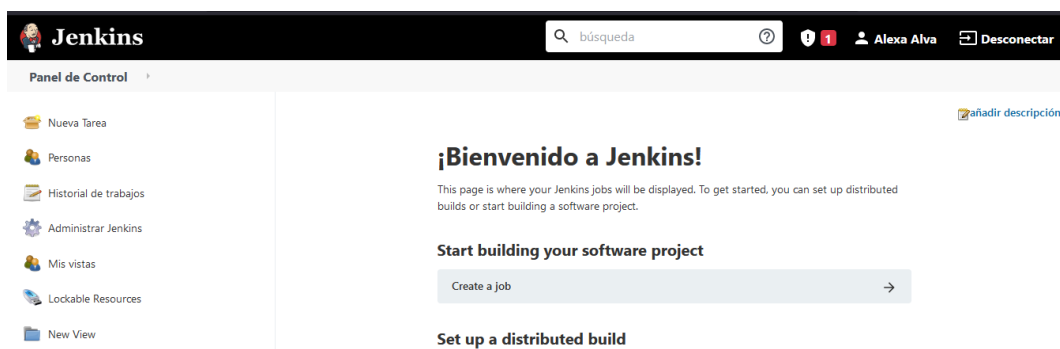
INTERFAZ

Ahora que vemos la interfaz de **Jenkins**, podemos encontrar en el menú superior derecho, los mensajes, la cuenta de usuario, y la opción de desconectar para finalizar sesión; y en el costado izquierdo, el menú de administración.

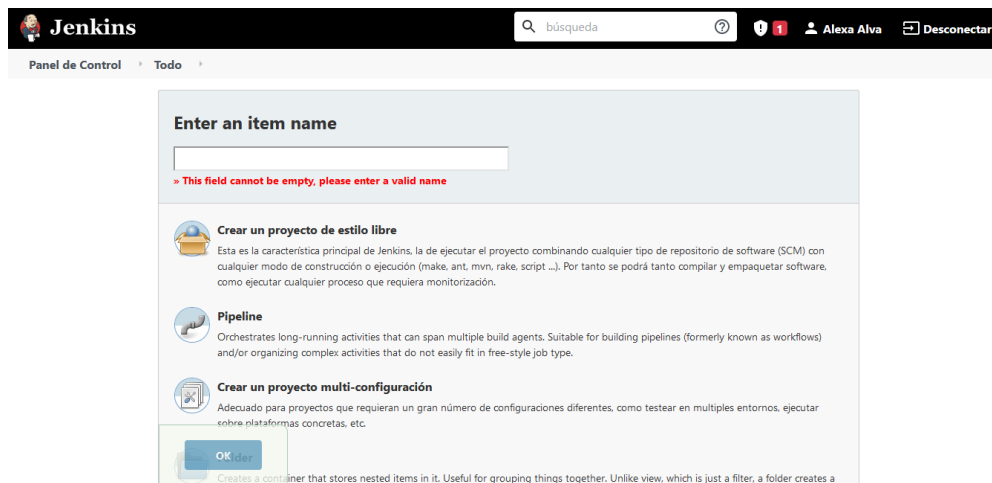


CREANDO EL PRIMER JOB

Ahora, aprenderemos cómo crear nuestra primera tarea, o **job**, para conectarnos a GitHub. Para ello, iremos al menú lateral izquierdo, y presionamos "Nueva tarea".

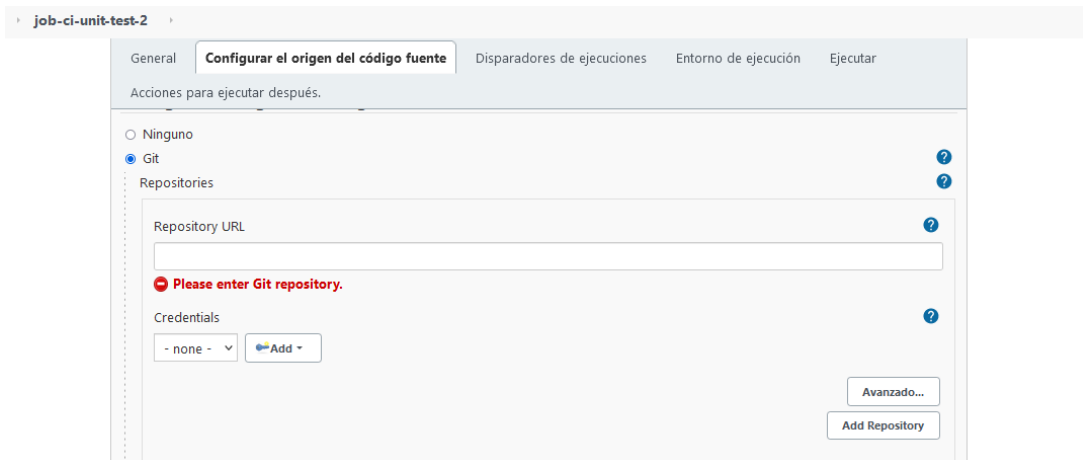


Llegamos a una nueva vista, donde hay que establecer un nombre de ítem. Le pondremos: **“job-ci-test-unit”**, que es muy parecido al nombre de repositorio al cual lo conectaremos. De inmediato seleccionamos la opción “crear un proyecto estilo libre”, y bajamos hasta encontrar el botón “OK”. Lo cliqueamos.



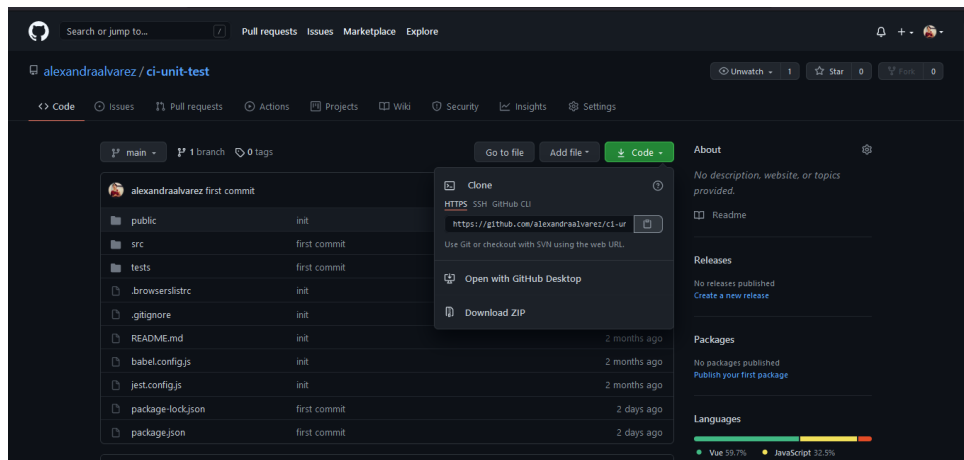
The image shows the Jenkins 'Enter an item name' dialog. At the top, there's a search bar and user information (Alexa Alva). Below the title, there's an input field for the item name. A red error message states: 'This field cannot be empty, please enter a valid name'. Below this, there are three options: 'Crear un proyecto de estilo libre' (selected), 'Pipeline', and 'Crear un proyecto multi-configuración'. The 'Crear un proyecto de estilo libre' option is highlighted with a blue border and an 'OK' button. Below the options, there's a small text description: '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'.

Llegamos a una nueva vista. Acá iremos a “Configurar el origen del código de fuente”, y seleccionaremos **“Git”**.

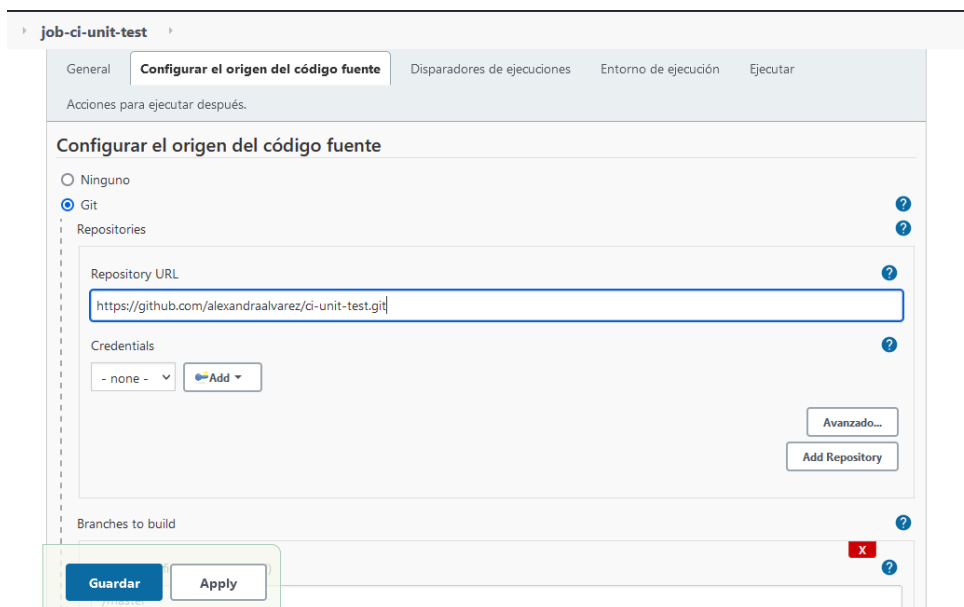


The image shows the Jenkins 'Configurar el origen del código fuente' dialog for the job 'job-ci-unit-test-2'. The 'General' tab is selected. Under 'Acciones para ejecutar después.', the 'Git' option is selected. Below this, there's a 'Repositories' section with a 'Repository URL' input field. A red error message states: 'Please enter Git repository.'. Below the URL field, there's a 'Credentials' section with a dropdown menu set to 'none' and an 'Add' button. At the bottom right, there are two buttons: 'Avanzado...' and 'Add Repository'.

Vamos a nuestra cuenta de **GitHub**, abrimos el repositorio que creamos al iniciar el CUE 4, y copiamos el enlace del repositorio.



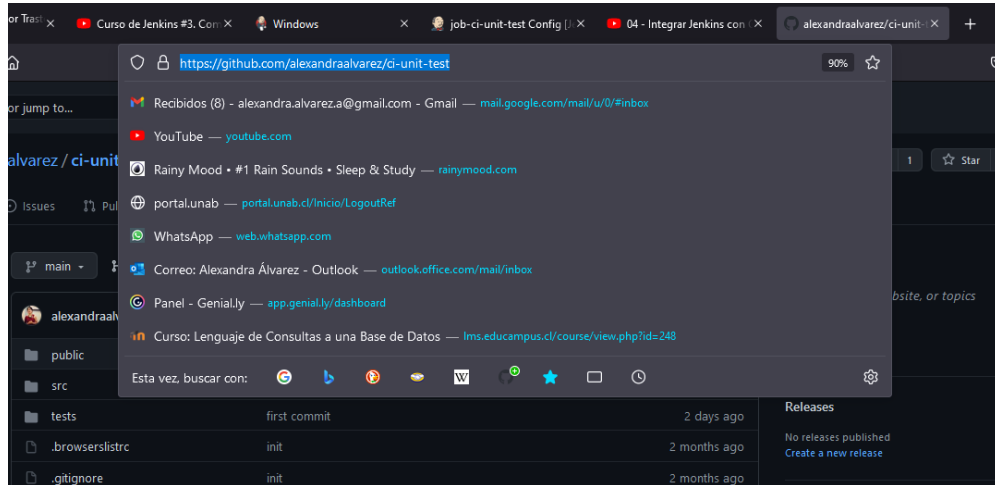
Lo pegamos en **Jenkins**, en el campo donde pide la **URL** del repositorio. Pasados unos segundos, vemos que desaparecen todos los mensajes en rojo, porque logró conectar con el repositorio.



Seguimos un poco más abajo, en **"Branches to build"**, y cambiamos el nombre de la rama que trae por defecto, de **"*/master"** a **"*/main"**, ya que en ésta guardamos nuestro proyecto. GitHub comenzó a utilizar el nombre **"main"**, en la rama principal de los proyectos creados a partir del día 1 de octubre del 2020. Esto pasa porque **"master"** hacía referencia al "amo", en la época de la esclavitud, un término considerado racista, pero en **Jenkins** aún no hacen la actualización, por lo que se configura manualmente.

Ahora, iremos a "General", y marcaremos **"GitHub project"**. Vemos que se abre una pestaña que nos pide agregar la **URL** del proyecto.

Nos dirigimos nuevamente a **GitHub**, y copiamos la **URL** desde la barra de navegación.



Y la pegamos en **Jenkins**.



Con esos pasos, ya logramos configurar **Jenkins**, para que apunte a nuestro repositorio previamente construido.

Ahora, le daremos las instrucciones para que esté atento, es decir, que **Jenkins** escuche, en un intervalo de tiempo, lo que sucede en nuestro repositorio. Para eso, iremos a “Disparadores de ejecuciones”, y seleccionaremos la opción “Consultar repositorio (**SCM**)”.



ontrol > job-ci-unit-test >

General Configurar el origen del código fuente **Disparadores de ejecuciones** Entorno de ejecución Ejecutar

Acciones para ejecutar después.

Disparadores de ejecuciones

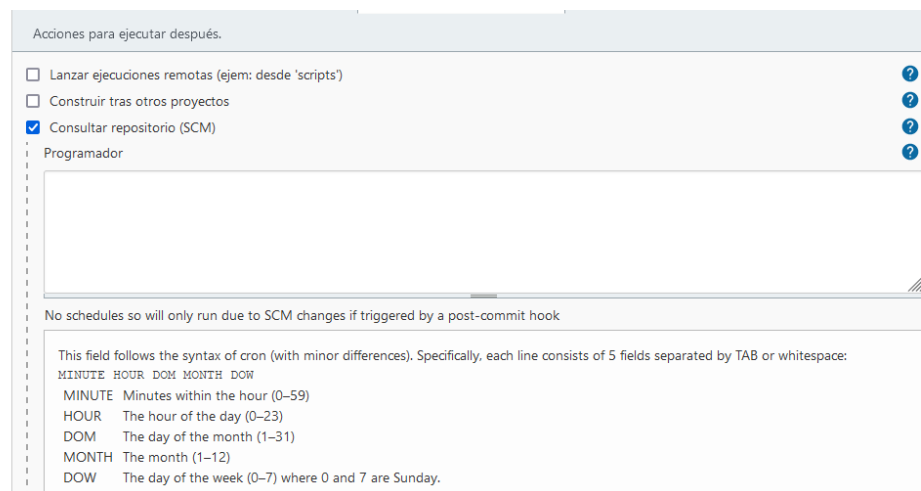
- ☐ Lanzar ejecuciones remotas (ejem: desde 'scripts')
- ☐ Construir tras otros proyectos
- ☒ Consultar repositorio (SCM)

Programador

No schedules so will only run due to SCM changes if triggered by a post-commit hook

- ☐ Ignore post-commit hooks
- ☐ Ejecutar periódicamente
- ☐ GitHub hook trigger for GITScm polling

Si pinchamos el botón de ayuda, que se encuentra justo en la esquina superior derecha del área de texto, podemos encontrar información que nos enseña la sintaxis para configurar el “Programador”.



Acciones para ejecutar después.

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


Programador

No schedules so will only run due to SCM changes if triggered by a post-commit hook

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)			
DOW	The day of the week (0–7) where 0 and 7 are Sunday.			

Vamos a escribir: **"H H 1,15 1-11 *"**, lo que se traduce en que el repositorio será revisado una vez al día, los 1 y 15 de cada mes del año, excepto en diciembre, y solo en el caso de que existan cambios, se ejecutará el **"job"**.



Acciones para ejecutar después.

Disparadores de ejecuciones

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

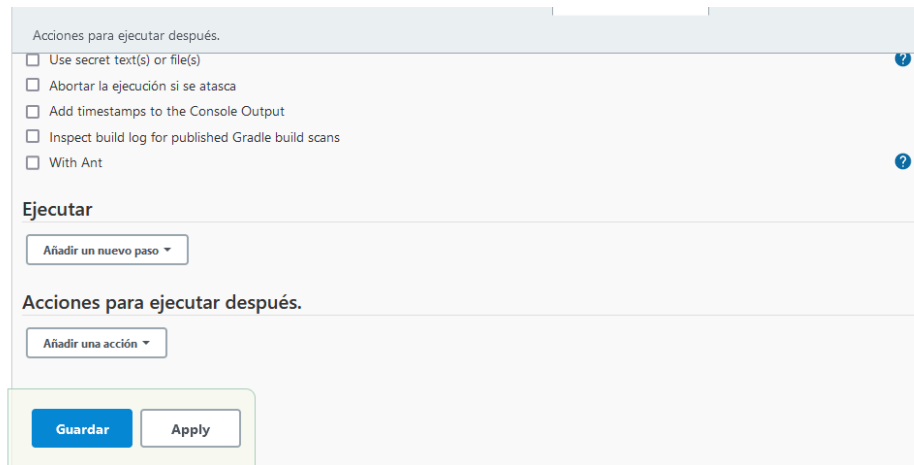
Programador

H H 1,15 1-11 *

Would last have run at martes, 15 de junio de 2021, 20:19:12 hora estándar de Chile; would next run at jueves, 1 de julio de 2021, 20:19:12 hora estándar de Chile.

- ☐ Ignore post-commit hooks
- ☐ Ejecutar periódicamente
- ☐ GitHub hook trigger for GITScm polling

Luego de eso, presionamos **"Guardar"**.



Acciones para ejecutar después.

- ☐ Use secret text(s) or file(s)
- ☐ Abortar la ejecución si se atasca
- ☐ Add timestamps to the Console Output
- ☐ Inspect build log for published Gradle build scans
- ☐ With Ant

Ejecutar

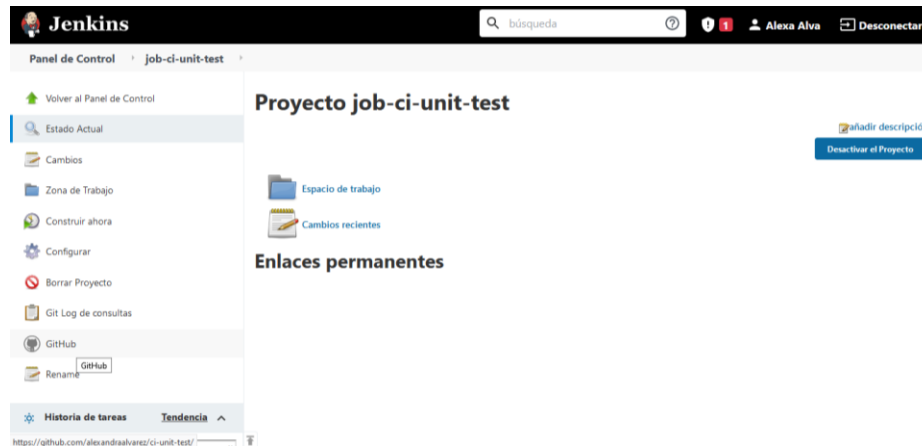
Añadir un nuevo paso ▾

Acciones para ejecutar después.

Añadir una acción ▾

Guardar Apply

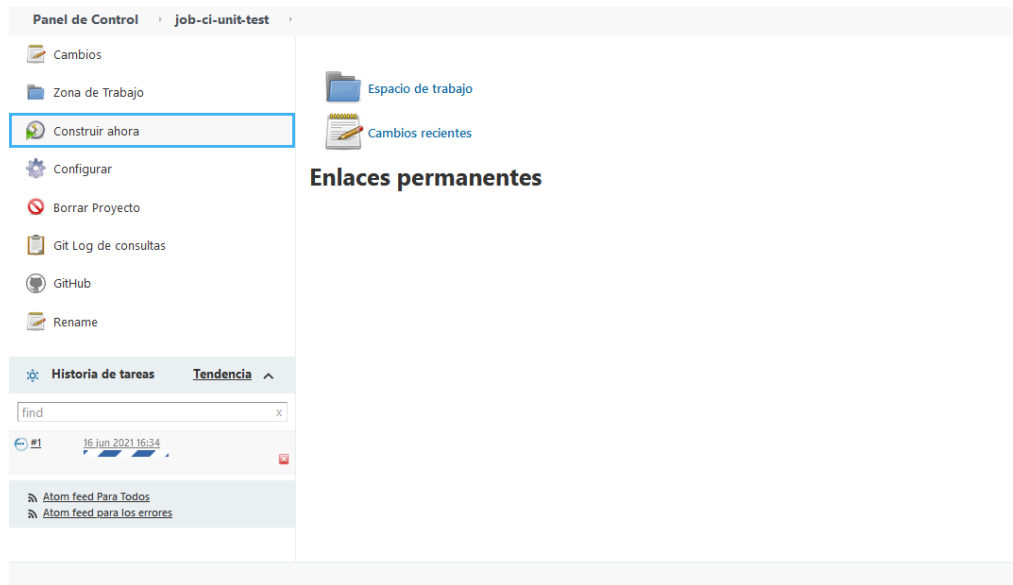
Y ya terminamos de crear la primera tarea. Ahora visualizaremos una nueva vista, donde, si miramos en el menú lateral izquierdo, podemos encontrar el símbolo de **GitHub**. Y si lo pinchamos, nos llevará a nuestro repositorio.



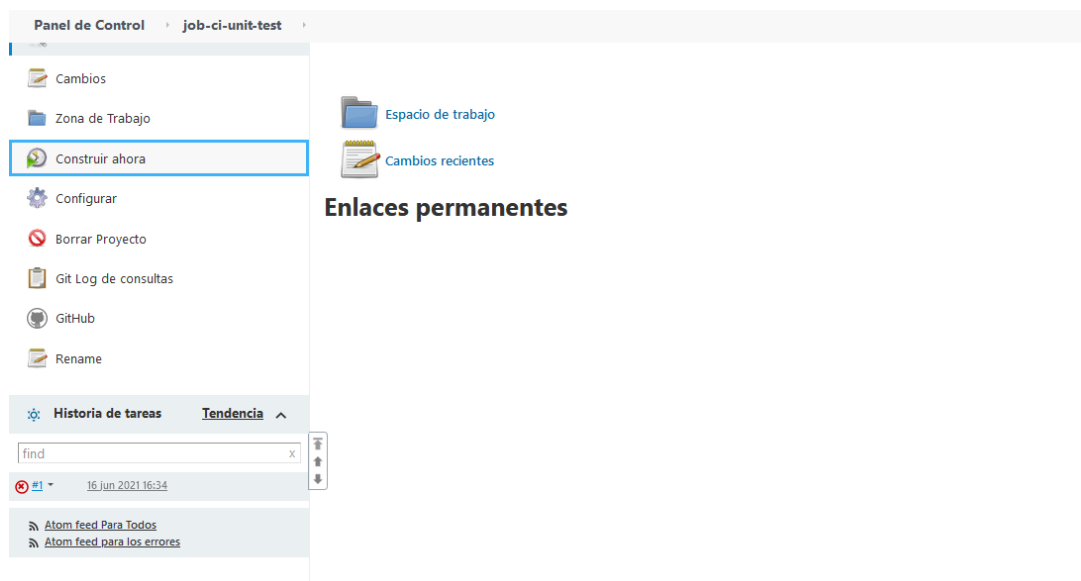
Si presionamos “construir ahora”, nos comenzará a crear las diferentes carpetas del proyecto, al interior de la carpeta **Jenkins**, pero primero nos enfocaremos en revisar la información que muestra la interfaz.



Al hacer clic, podemos ver cómo comienza a ejecutar, justo abajo del menú izquierdo, una barra de progreso.



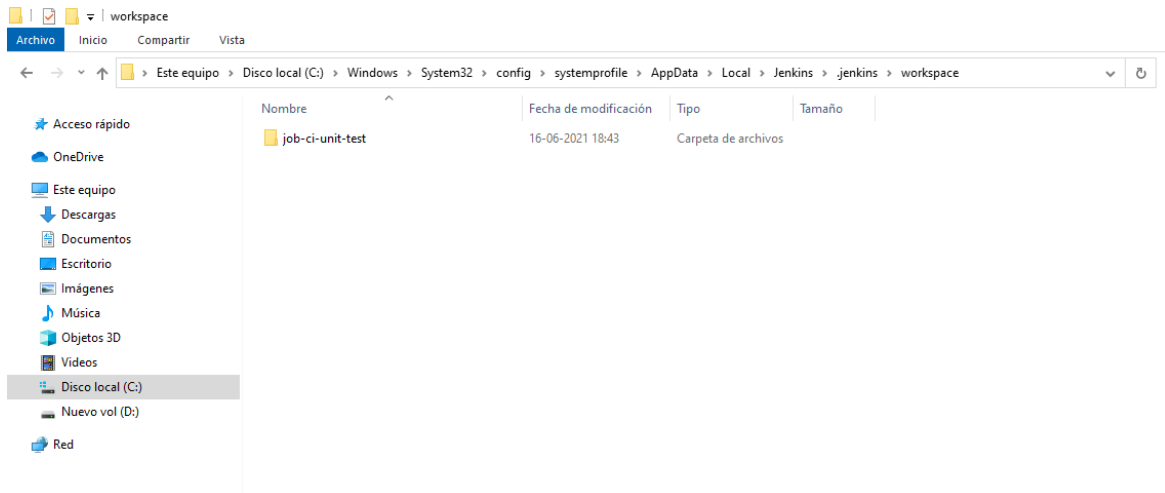
Cuando termina, podemos ir sobre el **#1** que aparece ahí, y pincharlo.



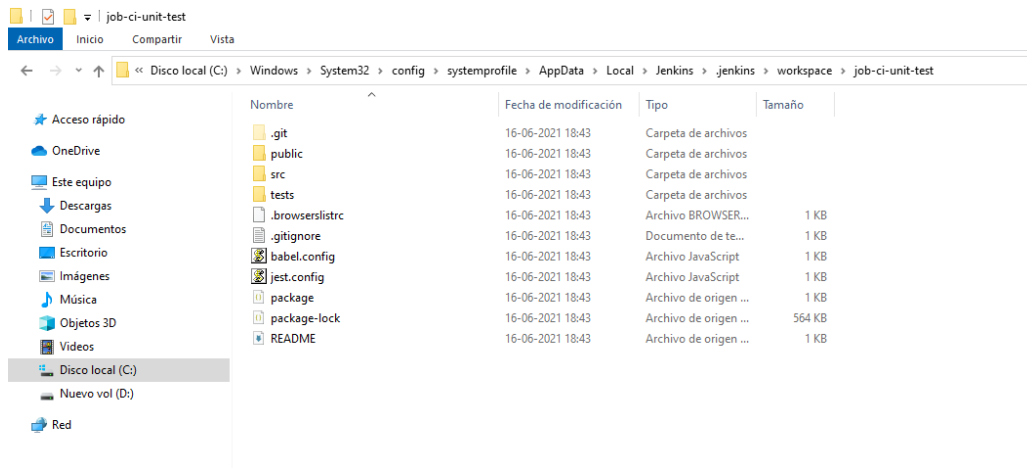
Notaremos que no hay cambios en nuestro proyecto.



También podemos revisar la salida de la consola, donde se ven todos los comandos que ejecutó. Entre ellos, se verá la ruta en la cual construyó nuestro proyecto en el computador. Si se puede notar, quedó en una ruta muy similar a la de la clave secreta de administrador.



Si entramos a esa carpeta, vemos que tenemos una réplica exacta de lo que subimos desde nuestro ejercicio.



Con esto, ya hemos aprendido a instalar y configurar **Jenkins**, para trabajar en equipos de desarrollo usando integración continua.