

Control de calidad

Instalación entorno de Testing en servidor Windows. [↗](#)

A continuación vamos a explicar los pasos necesarios para crear un entorno de testing en el servidor: MdesaTest (10.15.8.35).

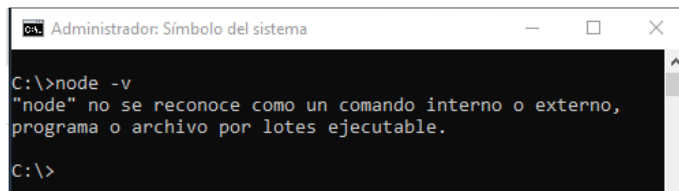
Tecnología necesaria:

- [Node.js 18.x o superior](#)
- [Nuclear Pumpkin Mayhem \(npm\)](#)
- [Cypress](#)
- [Mochawesome \(reports\)](#)
- [Jenkins](#)
- [Java JDK](#)
- [Extensión De Google Chrome Cypress Recorder](#)

Pasos:

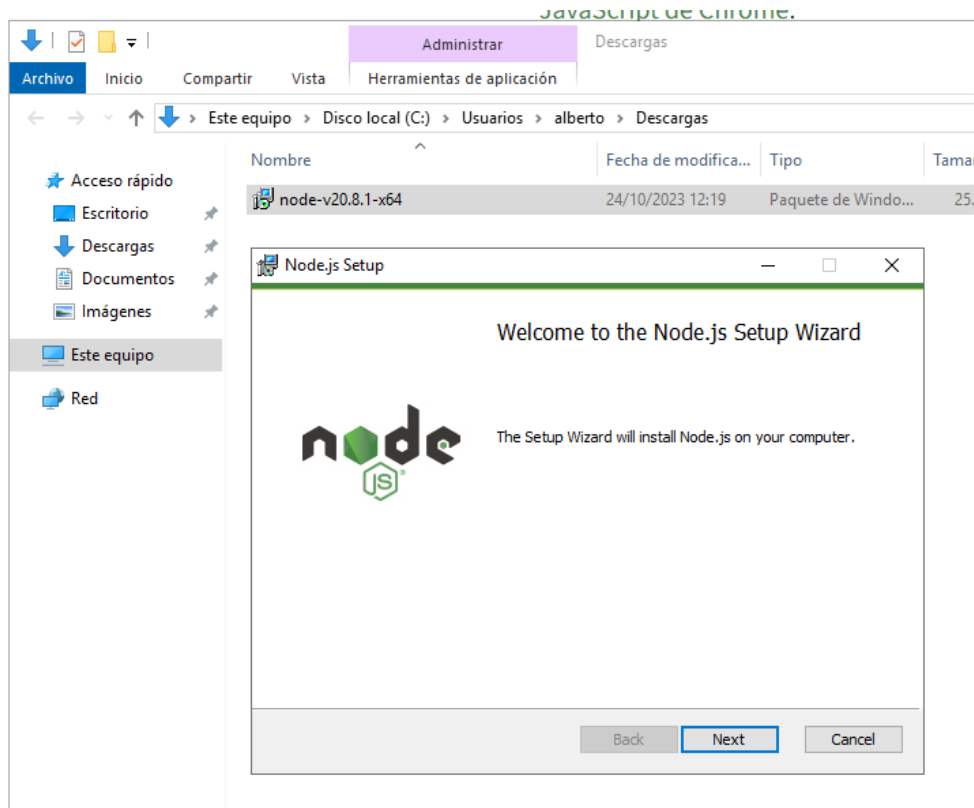
1. Abriremos una nueva consola de windows (CMD) como administrador y chequearemos qué versión de Node.js tenemos, si la hubiera, es necesario tener instalada una igual o superior a la 18.

Para ello pondremos la siguiente instrucción: **node -v** .



```
C:\>node -v
"node" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.
C:\>
```

Tanto si no tenemos instalado Node.js como si la versión que tenemos es inferior a la 18 debemos instalar [la ultima versión LTS de Node.js](#)



En este enlace descargaremos la última versión LTS que no es más que un archivo msi. La instalación que sea la más simple, le vamos dando next, next... y si nos pregunta que si queremos instalar “chocolatey”, no lo seleccionamos. Una vez terminada la instalación cerramos la consola de windows y la volveremos a abrir y volvemos a ver qué versión de Node.js tenemos, esta vez ya sí tenemos la última.

```
Administrador: Símbolo del sistema
C:\>node -v
v20.8.1
C:\>
```

2. Una vez instalado el Node.js pasamos a instalar [npm](#), para ello nuevamente en la consola de windows ejecutaremos la siguiente instrucción: **npm install -g npm@latest**. Una vez instalado, ejecutamos la instrucción **npm -v** para comprobar que la instalación ha sido correcta, se nos mostrará la versión de npm instalada.

```
Administrador: Símbolo del sistema
C:\>node -v
v20.8.1
C:\>npm install -g npm@latest
added 1 package in 5s
29 packages are looking for funding
  run `npm fund` for details
C:\>
```

3. Ahora pasamos a instalar Cypress que es una herramienta web destinada al testeo de componentes y webs end to end. Lo primero será crear un workspace en nuestro servidor , por ejemplo C:\Testing, una vez creado este workspace y situándonos en la carpeta

Testing ejecutaremos la siguiente instrucción en la consola de windows la instrucción: **npm install cypress --save-dev**

```
npm install cypress

C:\>node -v
v20.8.1

C:\>npm install -g npm@latest

added 1 package in 5s

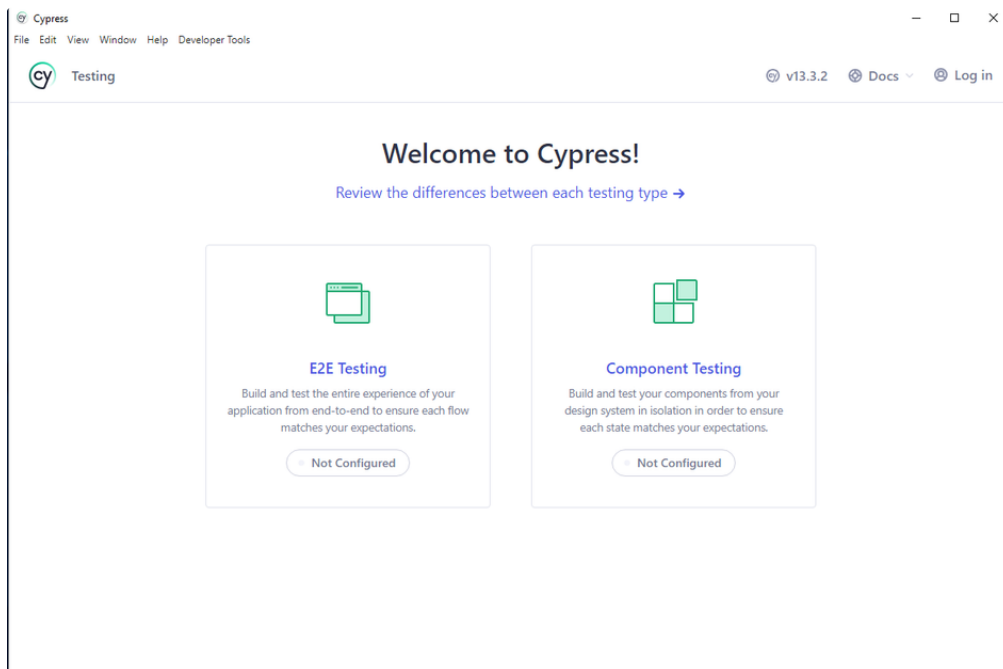
29 packages are looking for funding
  run `npm fund` for details

C:\>npm -v
10.2.1

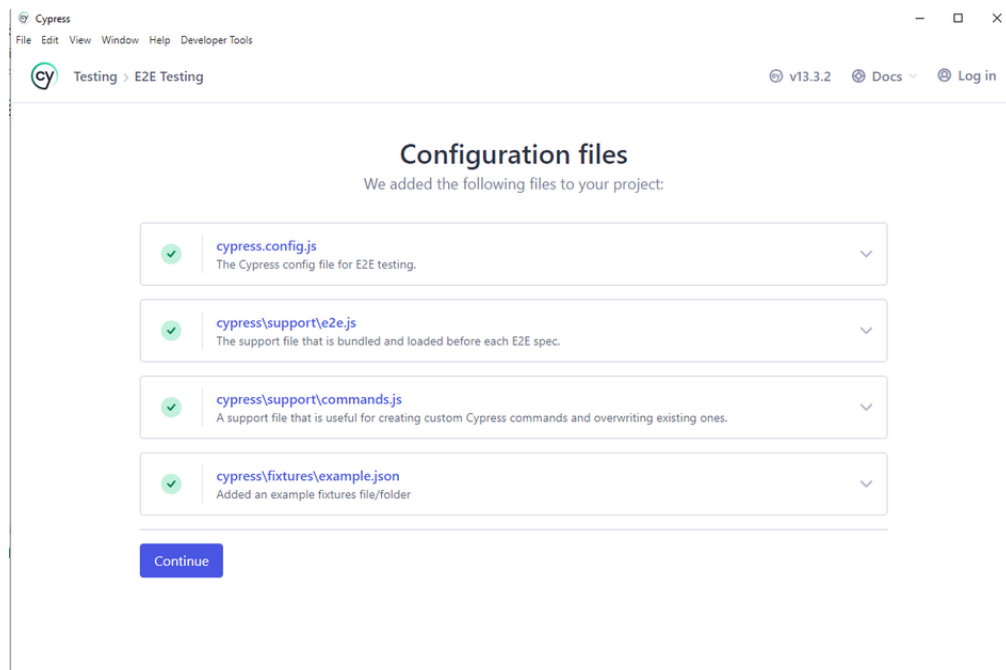
C:\>cd testing

C:\Testing>npm install cypress --save-dev
[ ] \ reify:@types/node: http fetch GET 200
```

4. Una vez que termine la instalación de Cypress, desde la carpeta Testing en consola de windows podemos ejecutar la instrucción: **npm run cypress open**, y se nos abrirá una ventana con el entorno de testing. Esto indica que la instalación ha sido correcta.



La primera vez que se inicia nos creará unos archivos de configuración de Cypress y unos ejemplos ya creados y unos archivos de configuración de Cypress necesarios para la instalación del Mochawesome, esos archivos se nos mostrarán en una nueva pantalla al hacer clic en E2E Testing, simplemente le daremos a Continue.



5. Una vez instalado Cypress pasaremos a instalar el componente [Mochawesome](#) encargado de crear los reportes de los resultados de los test en Cypress. Para ello pondremos en la consola de comandos, situándonos siempre en la carpeta Testing, la siguiente instrucción: **npm i --save-dev cypress-mochawesome-reporter**. Cuando haya terminado de instalar el componente tenemos que editar el archivo de configuración de Cypress, C:\Testing\cypress.config.js, y sustituir todo su contenido por este código:

```
1 const { defineConfig } = require("cypress");
2
3 module.exports = defineConfig({
4   reporter: 'cypress-mochawesome-reporter',
5   reporterOptions: {
6     charts: true,
7     reportPageTitle: 'custom-title',
8     embeddedScreenshots: true,
9     inlineAssets: true,
10    saveAllAttempts: false,
11  },
12  e2e: {
13    setupNodeEvents(on, config) {
14      // implement node event listeners here
15      require('cypress-mochawesome-reporter/plugin')(on);
16    },
17  },
18 });
```

Con esto terminaríamos de tener toda la parte de testing instalada y configurada con Cypress. Para ejecutar un test no visual y obtener su reporte en html debemos poner en la consola de comando la siguiente instrucción: **npm run cypress:run**.

Como es lógico para realizar la prueba tendremos que tener un test creado.

```
Administrador Símbolo del sistema
C:\Testing>npx cypress run

DevTools listening on ws://127.0.0.1:54765/devtools/browser/0bceec55-dd15-4f60-a274-31a0f96123d8

=====

(Run Starting)

Cypress:      13.3.2
Browser:      Electron 114 (headless)
Node Version: v20.8.1 (C:\Program Files\nodejs\node.exe)
Specs:        1 found (spec.cy.js)
Searched:     cypress/e2e/**/*.cy.{js,jsx,ts,tsx}

Remove output folder C:\Testing\cypress\reports\html

Running: spec.cy.js (1 of 1)

Adonline Test
Create Order
  ✓ Gets, types and asserts (40581ms)

1 passing (43s)

[mochawesome] Report JSON saved to C:\Testing\cypress\reports\html\.jsons\mochawesome.json

(Results)

Tests:      1
Passing:    1
Failing:    0
Pending:    0
Skipped:    0
Screenshots: 0
Video:      false
Duration:   42 seconds
Spec Ran:   spec.cy.js

Start generate report process
Read and merge jsons from "C:\Testing\cypress\reports\html\.jsons"
Enhance report
Create HTML report
HTML report successfully created!
C:\Testing\cypress\reports\html\index.html

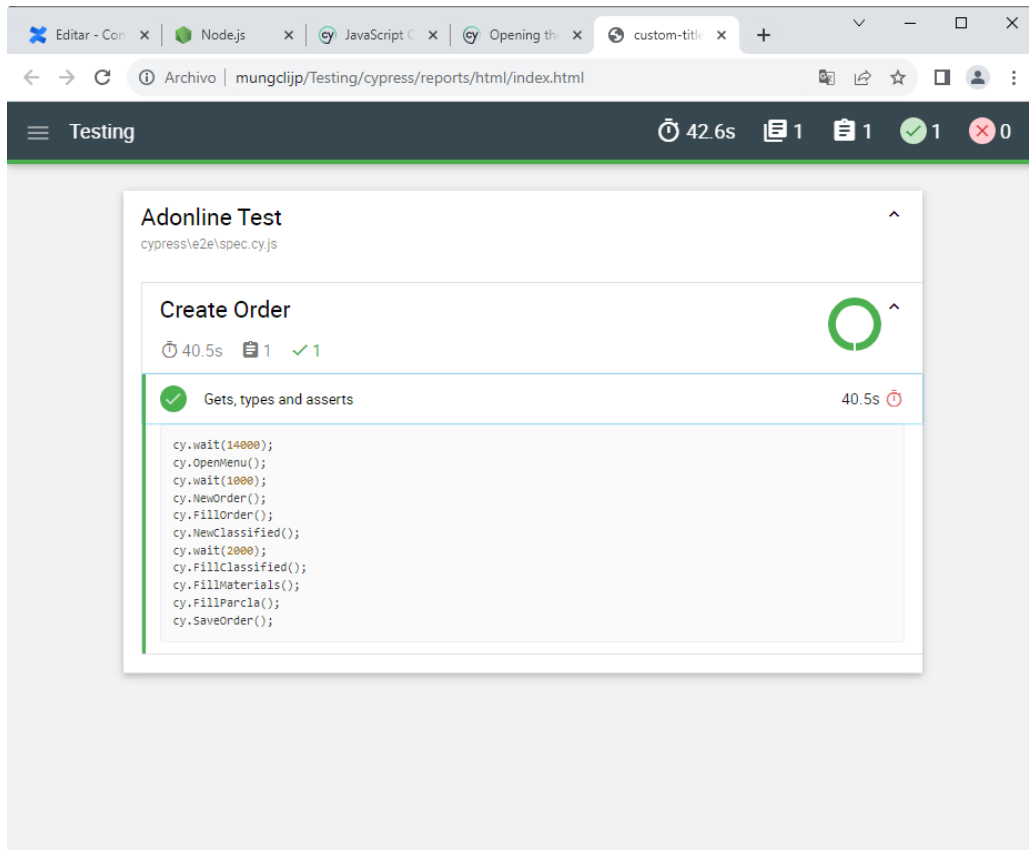
=====

(Run Finished)

Spec                                Tests  Passing  Failing  Pending  Skipped
┌──────────┬────────┬────────┬────────┬────────┬────────┬────────┐
│ ✓ spec.cy.js │ 00:42 │ 1      │ 1      │ -      │ -      │ -      │
├──────────┴────────┴────────┴────────┴────────┴────────┴────────┤
│ ✓ All specs passed! │ 00:42 │ 1      │ 1      │ -      │ -      │ -      │
└──────────┴────────┴────────┴────────┴────────┴────────┴────────┘

C:\Testing>
```

Al terminar de realizar el test, en la ruta C:\Testing\cypress\reports\html tendremos un archivo llamado index.html donde podremos ver todo la información del test realizado.



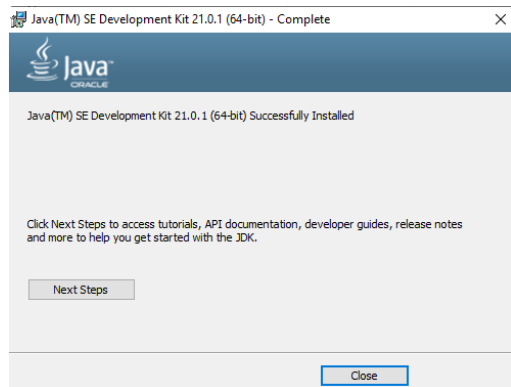
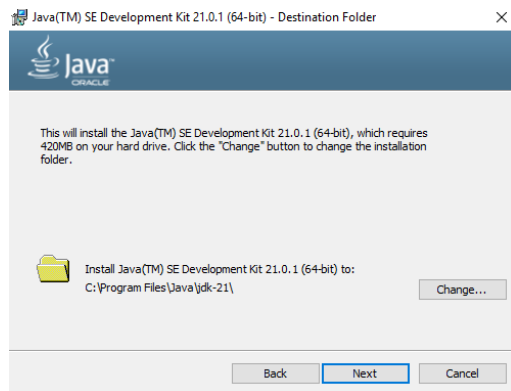
7. Jenkins es un servidor open source para la integración continua. Es una herramienta que se utiliza para compilar y probar proyectos de software de forma continua, lo que facilita a los desarrolladores integrar cambios en un proyecto y entregar nuevas versiones a los usuarios. Escrito en Java, es multiplataforma y accesible mediante interfaz web. Es el software más utilizado en la actualidad para este propósito.

Antes de hacer la instalación de Jenkins tenemos que comprobar si tenemos en nuestro sistema Java JDK, si no lo tenemos hay que instalarlo y si lo tenemos pero es una versión inferior a la 11 también tendremos que instalar una versión superior, [Java JDK](#), siempre se recomienda la última, pero en este caso con la que hemos testeado y funciona al 100% es con la 11.

Es posible que nos pida credenciales de oracle para descargar el jdk de Java, se puede crear una nueva cuenta o bien solicitar a IT una que usen ellos.

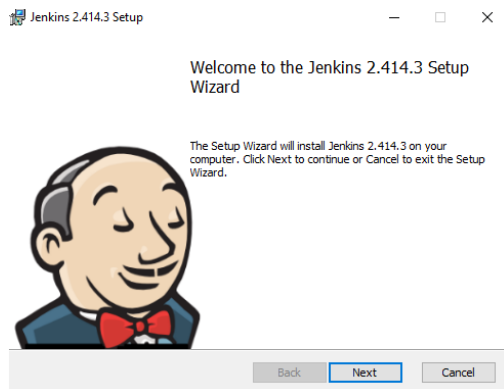
Los pasos a seguir son muy simples... next, next y next...



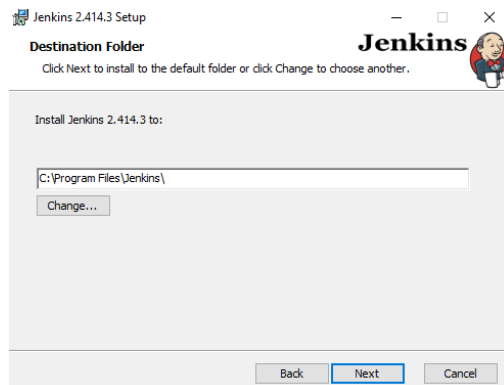


Una vez instalado el Java JDK el siguiente paso es instalar Jenkins y para ello seguiremos los siguientes pasos:

Lo primero que tenemos que hacer es descargarnos su archivo msi, [jenkins.msi](#), lo ejecutaremos y seguiremos la instalación.

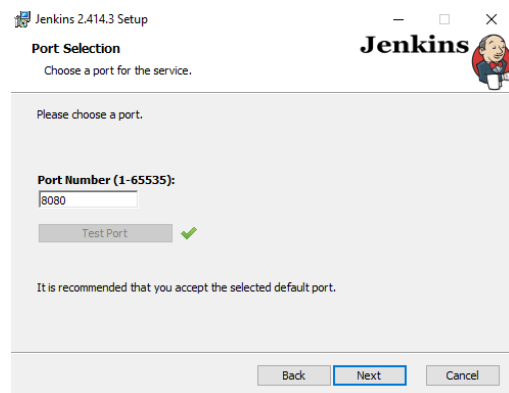


Seleccionaremos la ruta que viene por defecto de Jenkins para su instalación.

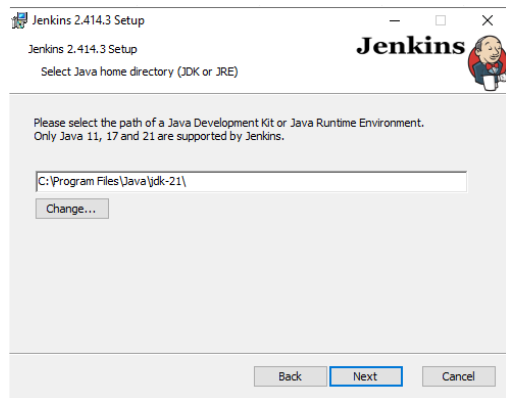


En esta ventana seleccionamos la primera opción “Run service as localSystem”.

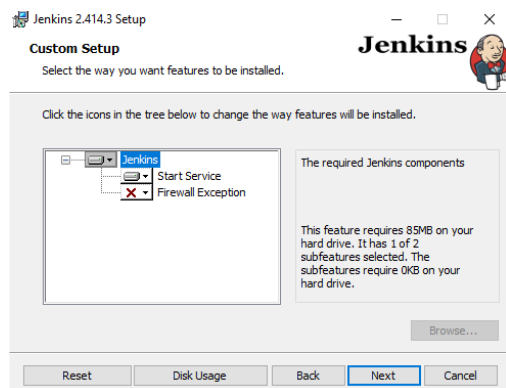
En la siguiente ventana seleccionaremos el puerto donde va a correr nuestro servidor de Jenkins, por defecto podemos seleccionar el :8080 tal y como nos sugiere la instalación.

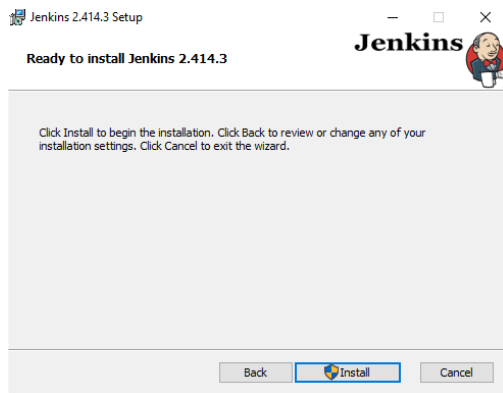


Ahora nos pide la ruta donde tenemos instalado en nuestro servidor el Java JDK, es por eso que necesitábamos instalarlo previamente.



En las dos siguientes ventanas le damos next, next y ya tendremos nuestro Jenkins montado en el servidor para poder, entre muchas otras acciones, programar nuestros test para que se ejecuten de forma automática.





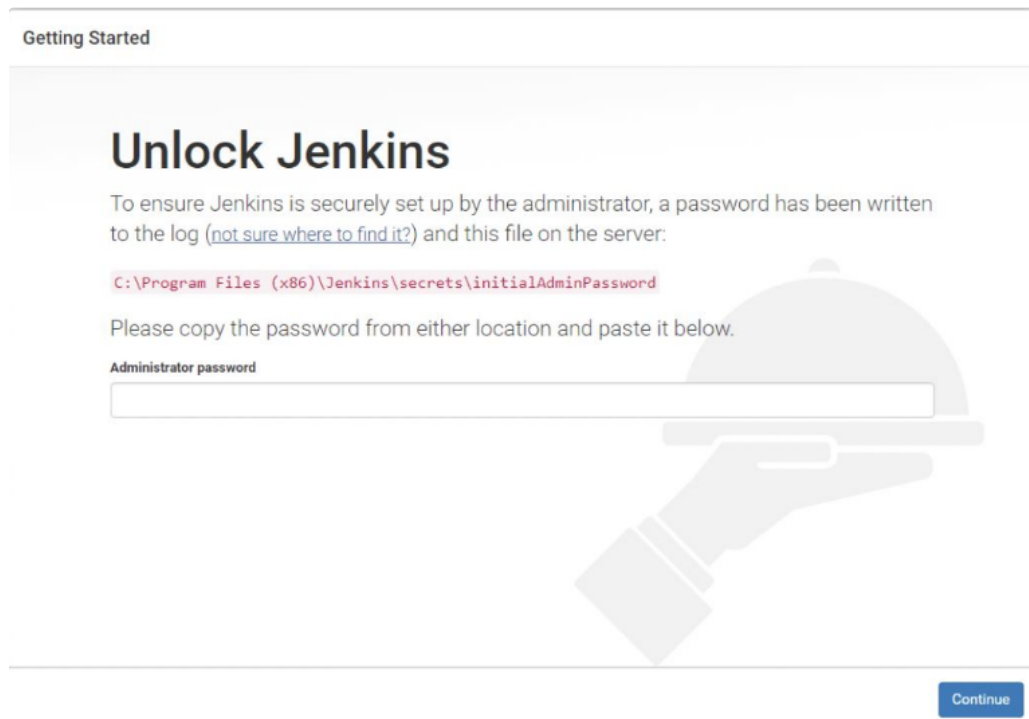
*Nota importante:

Es necesario que nuestro servidor tenga abierto el puerto :8080, en nuestro caso ya que es el que hemos configurado para que corra nuestro Jenkins, de no ser así será imposible entrar en su interfaz de usuario y configurarlo.

Para ejecutar Jenkins en nuestro servidor simplemente tendremos que escribir en nuestro navegador localhost seguido del puerto de acceso :8080.

<http://localhost:8080/>

Una vez instalado Jenkins la primera vez que lo ejecutemos nos va a pedir un password que debemos buscar en la siguiente ruta de instalación: C:\ProgramData\Jenkins\jenkins\secrets\InitialAdminPassword.



En la siguiente ventana nos va a preguntar qué plugins queremos instalar, aquí seleccionaremos los sugeridos por defecto.



En la siguiente ventana iremos viendo como se van instalando cada uno de los pluggins por defecto en nuestro sistema.

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	** Ionicons API
✓ Timestampers	Workspace Cleanup	Ant	Gradle	Folders
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View	OWASP Markup Formatter
Git	SSH Build Agents	Matrix Authorization Strategy	PAM Authentication	** Struts
LDAP	Email Extension	Mailer		** bouncycastle API
				** Instance Identity
				** JavaBeans Activation Framework (JAF) API
				** JavaMail API
				** Pipeline: Step API
				** Token Macro
				Build Timeout
				** Credentials
				** Plain Credentials
				** Trilead API
				** SSH Credentials
				Credentials Binding
				** SCM API
				** Pipeline: API
				** commons-lang3 v3.x Jenkins API
				Timestampers
				** Caffeine API
				** Script Security
				** JAXB
				** - required dependency

Jenkins 2.414.3

Cuando termine de instalar cada uno de los pluggins nos aparecerá la siguiente ventana donde pondremos nuestro usuario y password de administrador de Jenkins para la aplicación además del nombre completo y un correo electrónico.

Create First Admin User

Usuario

Contraseña

Confirma la contraseña

Nombre completo

Dirección de email

A continuación nos mostrará una nueva ventana con la url de acceso a Jenkins que ya sabemos

Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Y con estos pasos ya habríamos terminado de configurar Jenkins en nuestro equipo.

Jenkins is ready!

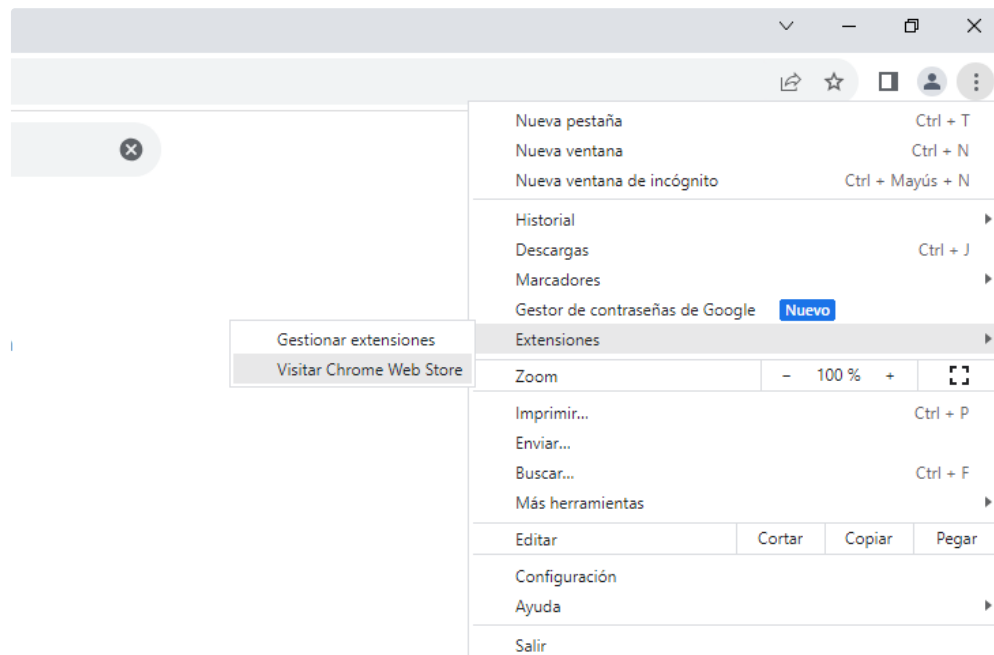
Your Jenkins setup is complete.

Start using Jenkins

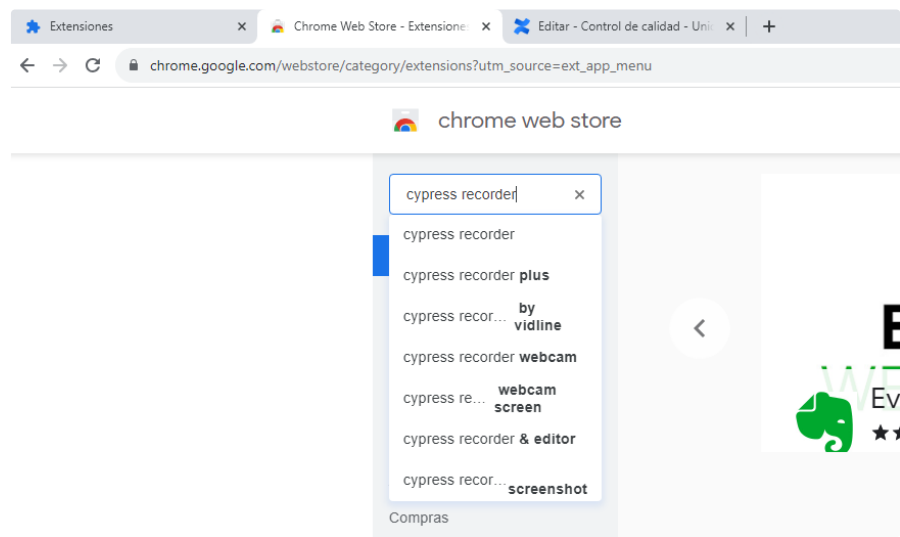
Jenkins 2.414.3

8. Por último instalaremos una extensión de Chrome llamada [Cypress Recorder](#) para poder hacer las grabaciones directamente desde nuestro navegador.

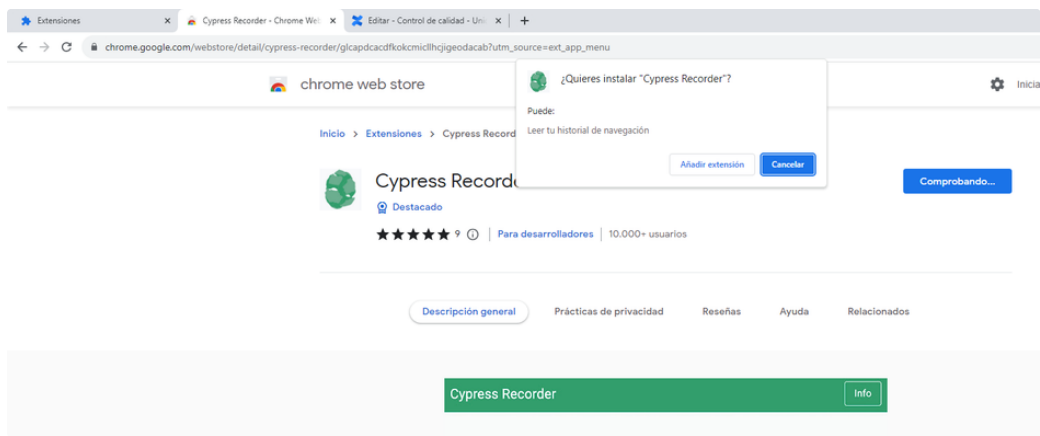
Lo primero que tenemos que hacer es irnos a la gestión de extensiones de Chrome en el propio navegador:



Una vez localizadas las extensiones buscaremos la que queremos instalar que en este caso será la de Cypress Recorder



Cuando la localicemos le daremos a instalar extensión



Por último le daremos a fijar en nuestro navegador para que en futuras ocasiones lo tengamos visualmente accesible en la barra de extensiones instaladas:

