## ¿Qué es Jenkins?

Jenkins es un aplicativo para la automatización del proceso completo de desarrollo de software, con tareas comunes como la integración continua pero sobretodo potenciando los equipos para implementar la parte técnica de la entregas continuas.

Para una persona que no está en contacto con el mundo tecnológico puede parecer un juego de palabras, pero básicamente esto viene a decir que disponemos de un software que nos permite automatizar los procesos de despliegue, la actualización de entornos, revisión de calidad de código y testing. Esto finalmente se traduce en entregas periódicas a cliente, en tiempo y forma con resultados que garantizan un producto funcional y listo para su explotación.

Jenkins debe su potencial a su amplio y versátil sistema de plugins que nos permiten realizar diferentes conjuntos de acciones, que permiten modularizar y extender la funcionalidad.

Descargar Jenkins (jenkins.war) de la página oficial.

<https://jenkins.io/>

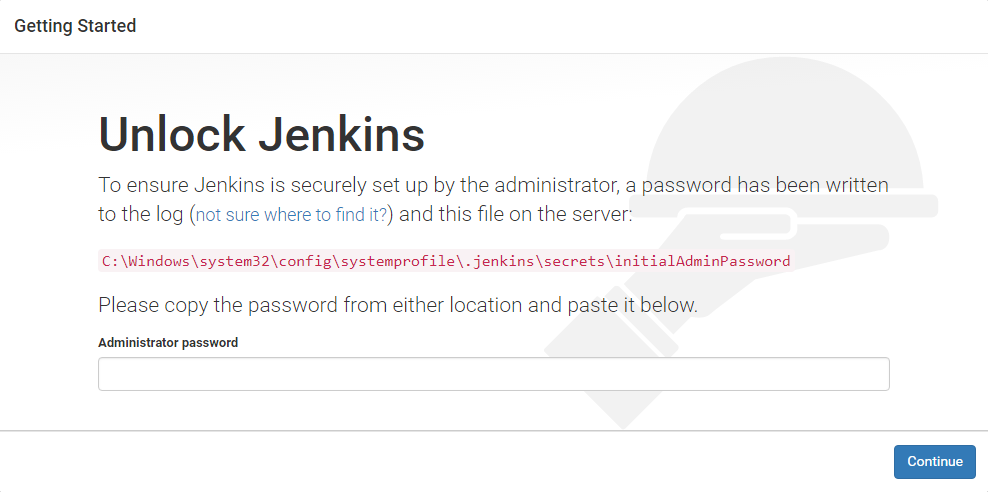
 https://updates.jenkins-ci.org/download/war/

Moverlo a la carpeta webapps de Tomcat

Levantar Tomcat

Cargar jenkins

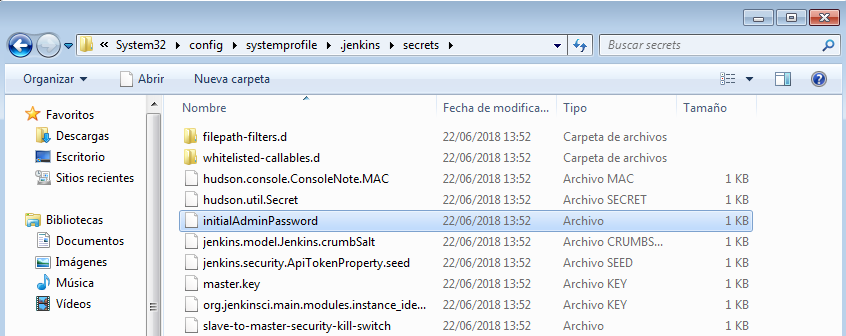


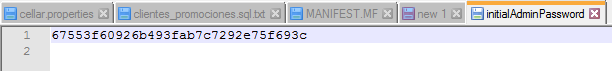


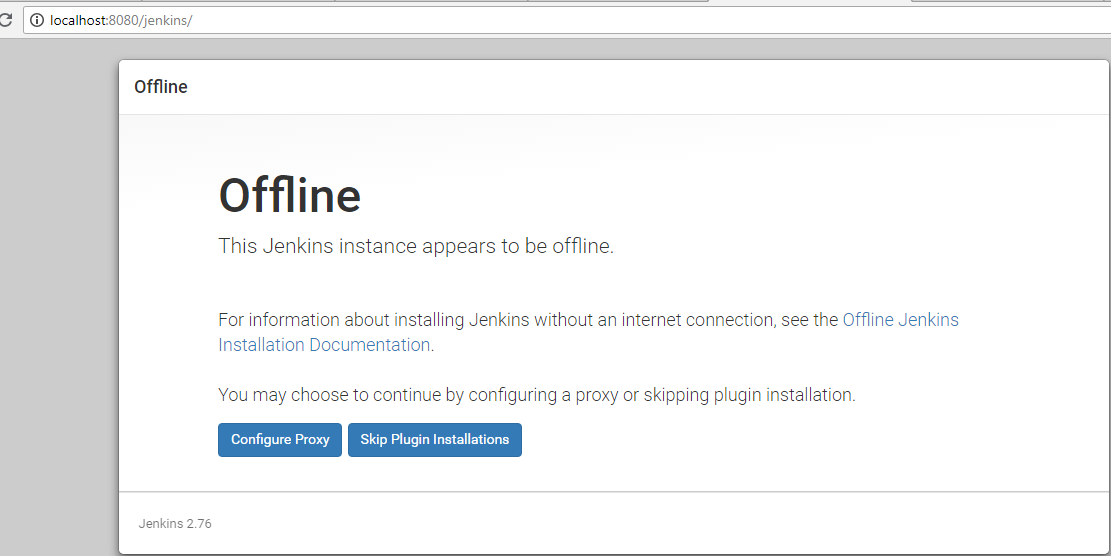
Administrator Password: Completar con la password que se genera en la carpeta

c:\windows\system32\config\systemprofile\.jenkins\secrets\initialAdminPassword

Abrir el archivo con un editor de texto y copiarlo en la caja de texto Administrator Password





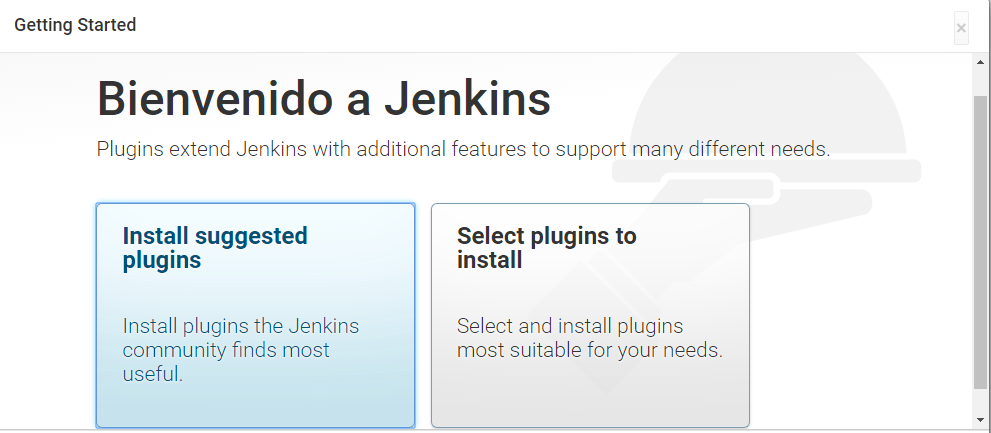


Configurar proxy en caso de necesitar un proxy con salida a Internet.

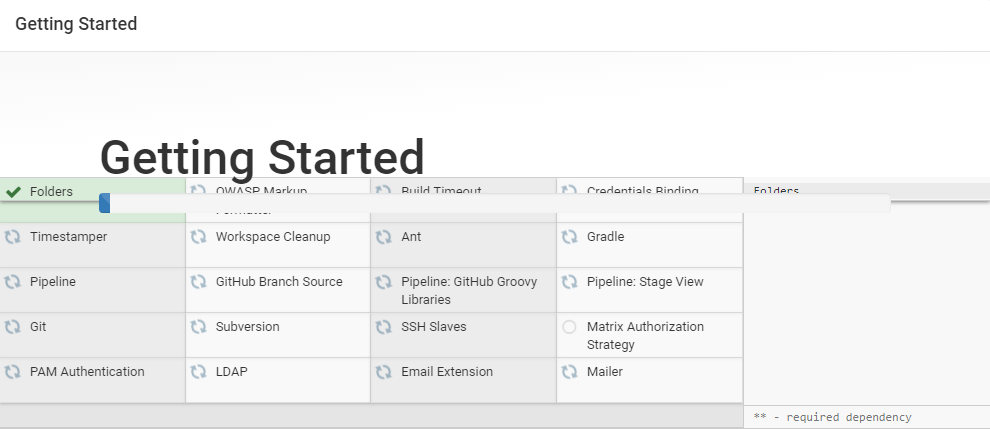
Saltear la instalación de plugins, se puede hacer mas adelante o bien instalar los plugins mas usados.

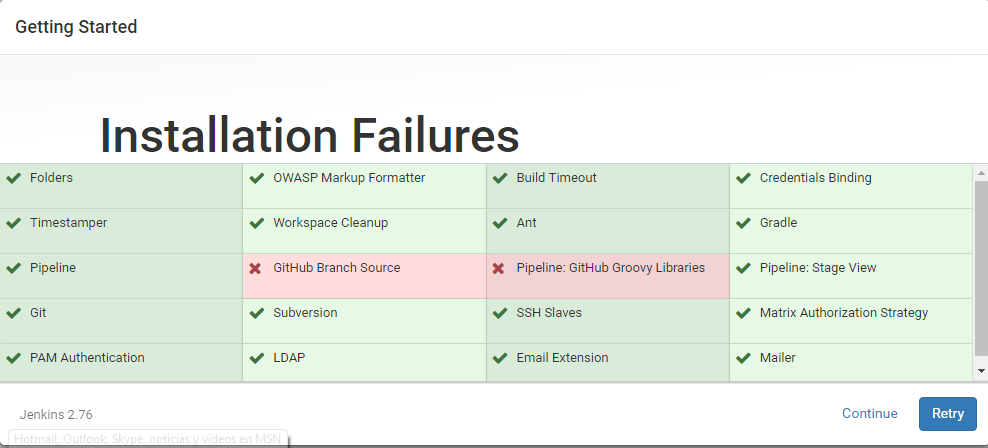
## ¿Qué son los plugins?

Jenkins está respaldado una gran comunidad, dónde cualquier desarrollador puede programar su propio plugin y ofrecer de nueva funcionalidad a nuestra aplicación de Jenkins. Algunos plugins realmente interesantes que usamos en nuestro día a día son FTP, SSH, GIT Conector, Task… existen cientos de plugins.

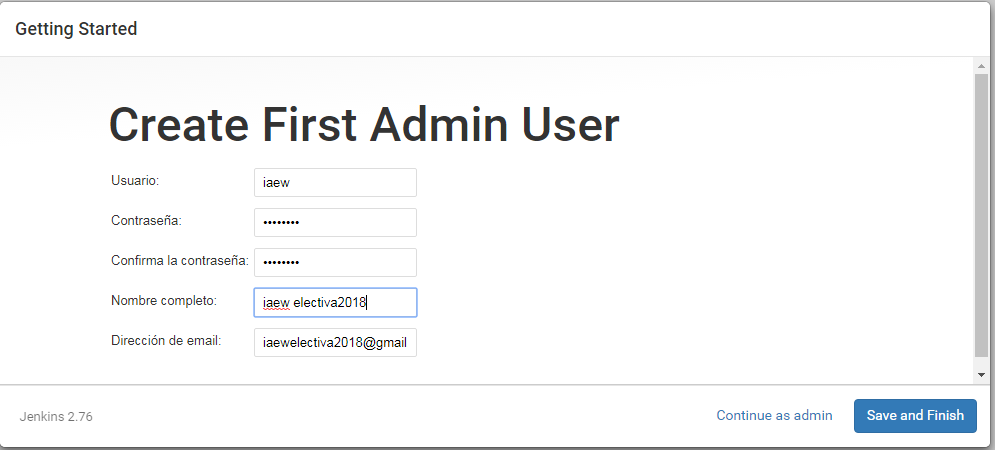


Seleccionar instalar los plugins sugeridos (los mas usados por la comunidad de Jenkins)





Continuar

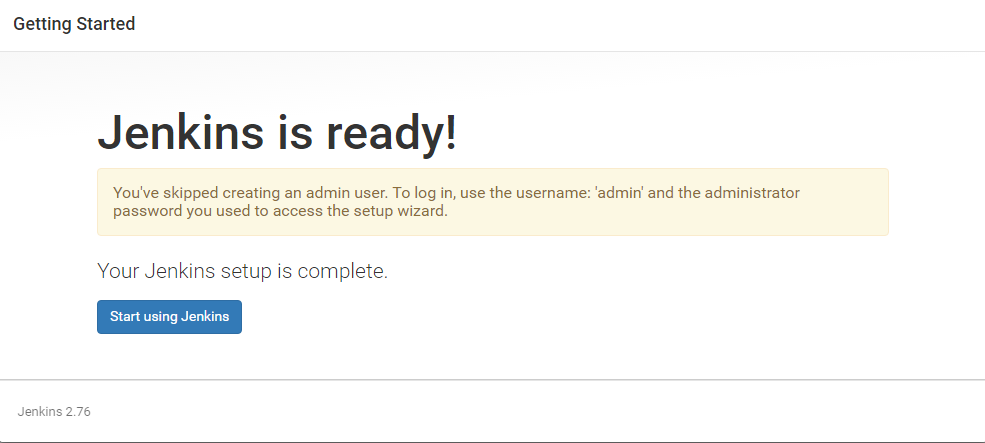


Creacion del primer usuario administrador

Usuario: iaew

Password: iaew2018

Continuar como admin

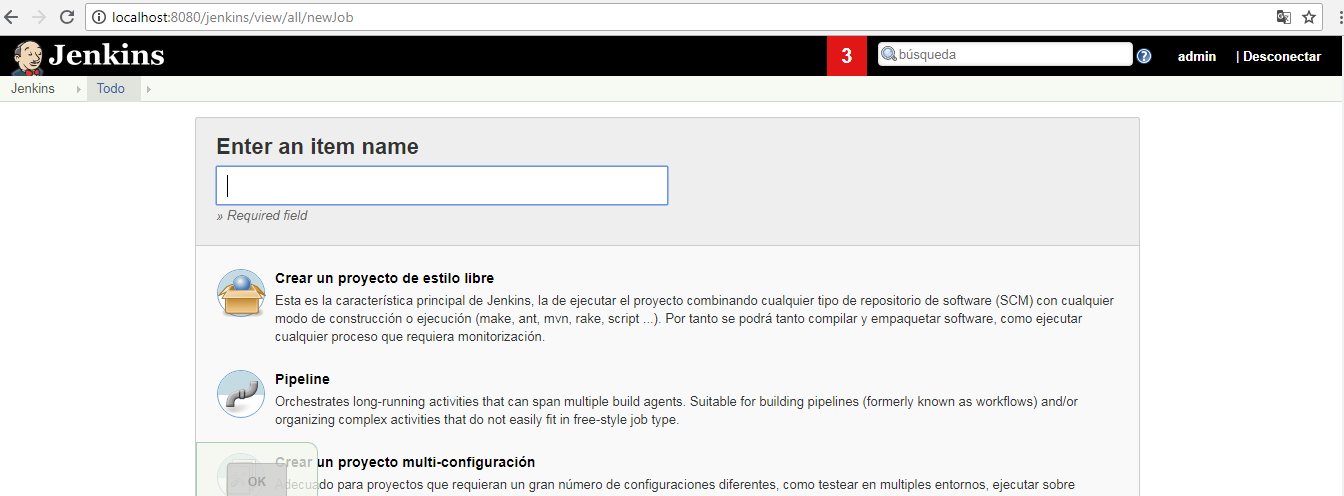


**Start Using Jenkins**

**Panel de Control**



**Crear una nueva tarea**



### Publicación de contenidos mediante FTP

Este [plugin](https://wiki.jenkins-ci.org/display/JENKINS/Publish+Over+FTP+Plugin) nos permite publicar contenido en una máquina dónde dispongamos de acceso mediante FTP. Este plugin es muy usado en máquinas dónde no se dispone de un usuario SSH con el que acceder a la máquina.

### Publicación de contenidos mediante SSH

Uno de mis preferidos por la potencia que nos brinda. Este [plugin](https://wiki.jenkins-ci.org/display/JENKINS/Publish+Over+SSH+Plugin) permite conectarnos a una máquina remota mediante SSH y realizar operaciones básicas como limpiar la cache de una aplicación, cambiar permisos a unas carpetas o ejecutar scripts en el servidor de destino.

### Control de calidad

Mas que a través de un plugin en concreto, el control de calidad y pruebas emplea toda una suite la cual nos permite realizar la ejecución de pruebas funcionales y unitarias y que valida cada entrega o cambio realizado a una aplicación .

Como complemento contamos con plugins de control de calidad de código, que nos dan brindan una serie de reportes basado en estándares de métricas que nos permiten aseverar que desarrollamos aplicaciones robustas y seguras .

## Gestión de usuarios

Otro aspecto muy importante a tener en cuenta a la hora de usar Jenkins es configuración o ejecución de tareas por usuario, lo que nos permite segmentar el acceso a los proyectos y entornos a los miembros autorizados.

## Satisfacción del cliente

Los errores humanos, por muy profesionales que sean los equipos, están a la orden del día. Jenkins nos permite que casi cualquier persona del equipo sea capaz de realizar un despliegue, interpretar los resultados obtenidos y generar un producto listo para entregar al cliente.

Pondremos un ejemplo de un entorno algo complejo: se ha desarrollado una nueva funcionalidad que se encuentra lista para pasar a producción. Bajo este escenario  una vez concluidas todas las pruebas se notifica a nuestro equipo de sistemas, los cuales recibirán un paquete de actualización con los cambios a propagar a producción, y un listado de tareas a ejecutar: actualización de bases de datos, limpiar memoria del sistema, limpiar caches de aplicación y servidor, actualizar nodos dónde se tiene la aplicación replicada para mejorar la tasa de respuestas, etc. Como véis no son tareas triviales, y saltarnos cualquiera de estos pasos puede terminar en una aplicación caída en un entorno de producción que pudiera ser un comercio electrónico grande o una página de turismo con millones de visitas mensuales.

Con Jenkins todas estas tareas se planifican y configuran desde el minuto 0 incluyendo contramedidas si cualquiera de ellas falla, reduciendo al mínimo la ocurrencia de errores, los problemas de despliegue y el downtime entre actualizaciones.

## Control de lo que desplegamos

A veces es difícil mantener informado al cliente de todo lo que va a recibir en un sprint o entrega. Con Jenkins y su integración con Git (el sistema de control de versiones que empleamos), este reporte es ta sencillo como darle al botón de **Ejecutar tarea** y obtendremos los cambios realizados con respecto al despliegue anterior. Hay que recalcar que en este aspecto es importantísimo que nuestro desarrolladores realicen trabajos atómicos, que no mezclen funcionalidades en cada trabajo que realizan y que se sigan algunas recomendaciones sobre escritura de mensajes. Estos mensajes finalmente pueden ser extraidos por el encargado de despliegues o el jefe de proyecto y se emplean para la elaboración de un documento más formal para entregar a los clientes con las tareas realizadas desde la última versión del aplicativo.

<https://www.hiberus.com/crecemos-contigo/jenkins-como-automatizar-tareas-despliegues-testing/>

<http://www.javiergarzas.com/2014/05/jenkins-en-menos-de-10-min.html>

<http://www.robertocrespo.net/kaizen/aprende-a-montar-un-entorno-de-integracion-continua-v-ejecucion-de-jobs-en-jenkins/>

Ejercicio: Descargar un proyecto git

[](http://52.7.186.71/jenkins/)

Instalar git

apt-get install git

Pide loguear como root

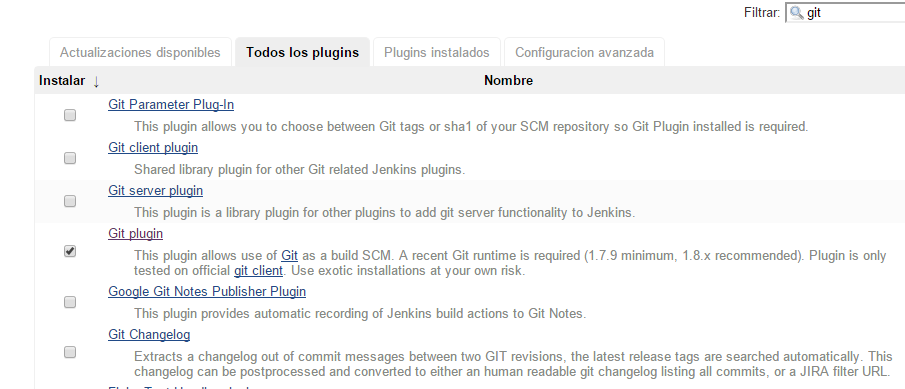
sudo passwd root (cambiar contraseña de root)

su root

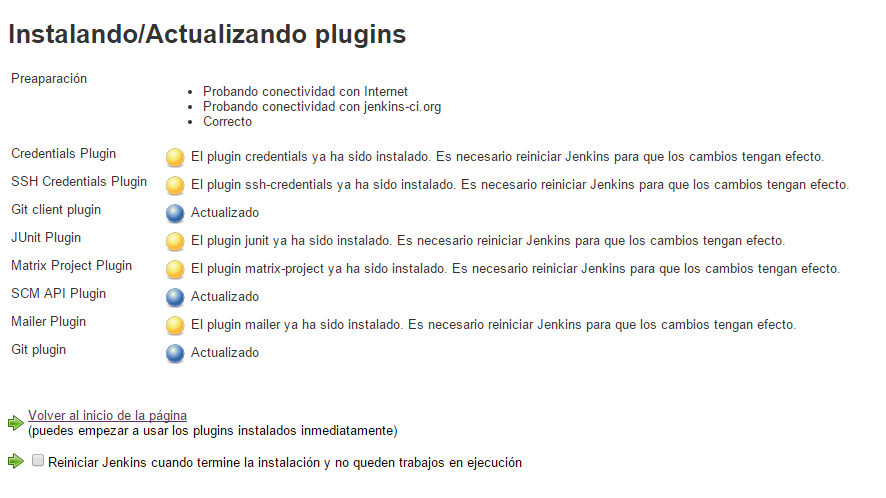
psw: \*\*\*\*\*\*

Configurar Git plugin

Git no viene por defecto, es necesario instalar el plugin de Git para Jenkins. Ir al menú “Administrar Jenkins” y seleccionar “Administrar Plugins” - “Todos los plugins”. Filtrar por “git plugin” y seleccionar para su instalación. Pulsar el botón “Instalar sin reiniciar”.



Seleccionar Instalar sin reiniciar



Una vez instalado reiniciar Jenkins para que tengan efecto.

Creé un repositorio en GitHub para hacer la prueba de configurar una tarea

<https://github.com/reedcons/pruebas_gradle.git>

#### Ir a Jenkins- configurar el sistema

#### 

#### Indicar el path a git

#### 

#### Crear las credenciales o tokens

#### 

#### Crear la tarea que descarga de git

#### Crear proyecto libre:

#### Configurar donde se encuentra el fuente

#### 

#### 

#### Salida de la tarea:

Cloning the remote Git repository

Cloning repository <https://github.com/reedcons/pruebas_gradle.git>

> git init /root/.jenkins/jobs/proy\_libre/workspace # timeout=10

Fetching upstream changes from <https://github.com/reedcons/pruebas_gradle.git>

> git --version # timeout=10

> git -c core.askpass=true fetch --tags --progress <https://github.com/reedcons/pruebas_gradle.git> +refs/heads/\*:refs/remotes/origin/\*

> git config remote.origin.url <https://github.com/reedcons/pruebas_gradle.git> # timeout=10

> git config --add remote.origin.fetch +refs/heads/\*:refs/remotes/origin/\* # timeout=10

> git config remote.origin.url <https://github.com/reedcons/pruebas_gradle.git> # timeout=10

Fetching upstream changes from <https://github.com/reedcons/pruebas_gradle.git>

> git -c core.askpass=true fetch --tags --progress <https://github.com/reedcons/pruebas_gradle.git> +refs/heads/\*:refs/remotes/origin/\*

> git rev-parse refs/remotes/origin/master^{commit} # timeout=10

> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10

Checking out Revision 97351b820b71afe6e44b05d1a6aed9f052be7304 (refs/remotes/origin/master)

> git config core.sparsecheckout # timeout=10

> git checkout -f 97351b820b71afe6e44b05d1a6aed9f052be7304

First time build. Skipping changelog.

Finished: SUCCESS

#### Construir Ahora: Ejecuta al instante el Proyecto

#### 