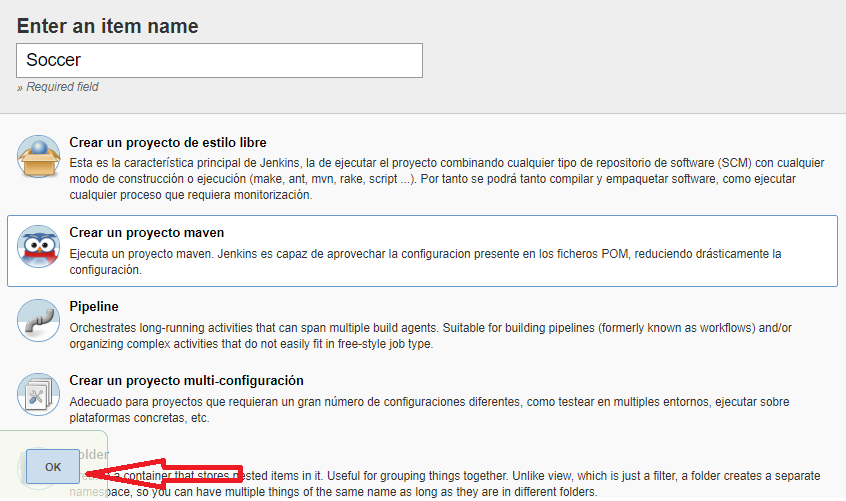
**Nuestra primera tarea**

Como hemos comentado, existen distintos **tipos de tareas** que podemos crear desde **Jenkins**. Para esta primera tarea, he elegido un caso básico en el que tendremos un **proyecto Maven**, subido a un **repositorio GitHub**, y que controlaremos desde **Jenkins**. Accedamos en primer lugar a la pantalla de creación de nueva tarea (pulsar sobre **“Nueva tarea”**).

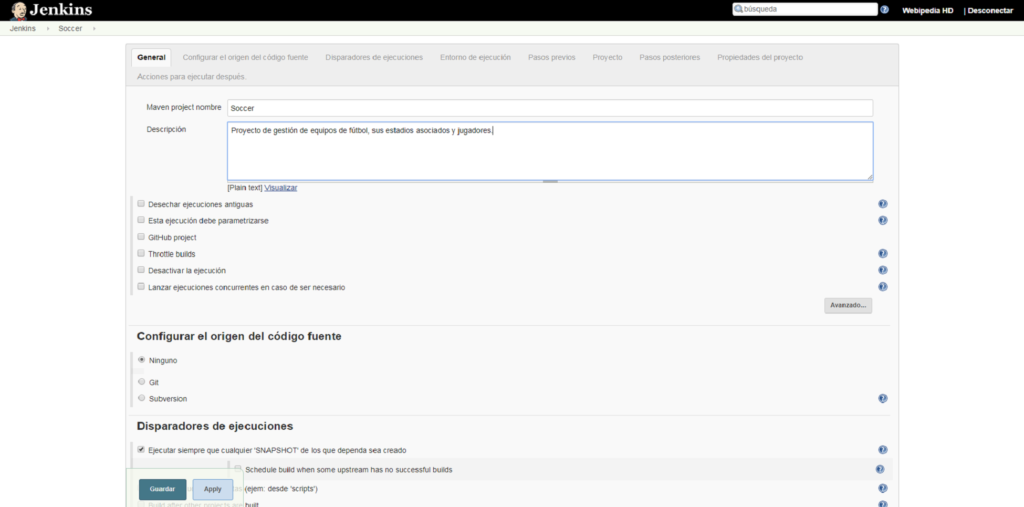


Voy a utilizar como base un proyecto muy sencillito llamado **“Soccer”**, que permite la gestión de equipos de fútbol, los estadios asociados y los jugadores de cada equipo. Por tanto, llamaré a la tarea por el mismo nombre (es decir, **“Soccer”**). Seleccionamos después la opción **“Crear un proyecto maven”**, con lo que se habilitará el botón **“OK”** de la parte inferior (lo pulsamos).

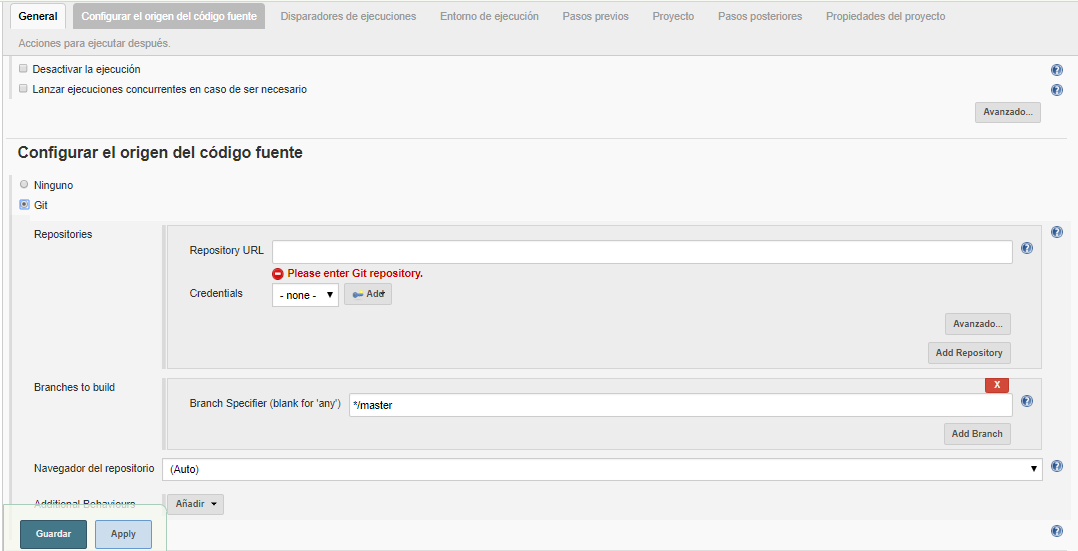


Y pulsamos el botón OK

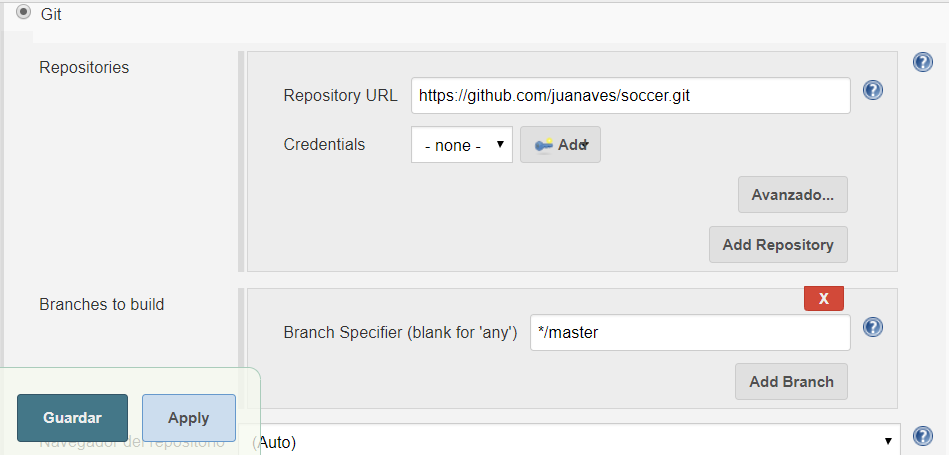
Ésta es la **pantalla de configuración** de nuestra tarea. Lo primero que haremos será poner una **descripción** acorde con la naturaleza de nuestro proyecto.



Desde la sección **“Configurar el origen del código fuente”**, indicamos que la herramienta utilizada es **“Git”**, con lo que se desplegará la sección de configuración correspondiente.

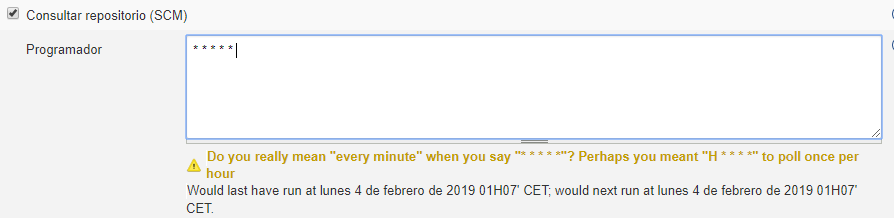


Lo único que debemos hacer aquí es configurar nuestro repositorio en **GitHub**. En mi caso, teclearé **“https://github.com/juanaves/soccer.git”**. Al tratarse de un **repositorio público**, no es necesario ningún otro tipo de configuración de **credenciales** extra. Para verificar que todo está correcto, basta con pulsar sobre cualquier otro lugar fuera del área de edición y, si **Jenkins** ha accedido correctamente a la URL indicada, desaparecerá el texto de aviso en rojo.



Pasamos ahora la sección **“Disparadores de ejecuciones”**. Una configuración típica de Jenkins (y es aquí donde vemos la verdadera potencia de Jenkins) consiste en programar la herramienta para que, cada vez que se realice una **actualización del repositorio** de nuestro proyecto, **Jenkins** automáticamente coja el  **código fuente** del proyecto y lo  **compile/despliegue** de nuevo.

Para habilitar este **pooling**, marcamos el check **“Consultar repositorio (SCM)”**. Pulsando sobre el símbolo de interrogación de la derecha  accederemos **a información detallada** sobre las opciones de configuración disponibles, las cuales determinarán la **frecuencia** con la que Jenkins observará variaciones en nuestro repositorio. Por ejemplo, si queremos que se realice cada hora, teclearemos **“H \* \* \* \*”**, si queremos que se realice cada 15 minutos teclearemos **“H/15 \* \* \* \*”**, etc. En nuestro caso, queremos un refresco más frecuente, introduciremos cinco asteriscos **separados por espacios**. Esto hará que se consulte el repositorio para comprobar si ha habido cambios todos los minutos de todas las horas de todos los días.

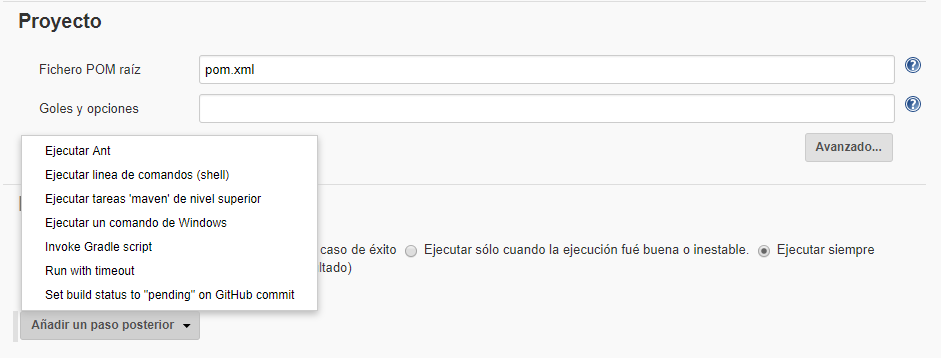


Desde la sección **“Proyecto”** vemos que se indica **“pom.xml”** como fichero POM raíz. Nuestro proyecto en GitHub contiene dicho POM en el directorio raíz, así que aquí **NO tenemos que tocar nada.**

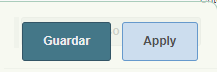


Por último, desde la sección **“Pasos posteriores”** podemos indicar algún **paso posterior** a realizar una vez compilado el proyecto, como por ejemplo copiar el “.war” generado a otro directorio, borrar ficheros temporales de algún directorio, etc.

**Lo dejamos como esta**.



Ya estaría todo. Pulsamos sobre **“Guardar”**.



**Compilación de proyectos**

Una vez configurado nuestro proyecto, **llega el momento de compilarlo**. Para ello, pulsaríamos en **“Construir ahora”**.

En cualquier caso, sin que tengamos que pulsar nada, Jenkins realiza una **“construcción inicial”** de nuestro proyecto. En la sección **“Historia de tareas”** de la parte inferior izquierda, veremos el listado de construcciones lanzadas. En nuestro caso, aparece una única construcción, nombrada como **“#1”** (la bolita de color azul indica que la tarea se ha ejecutado correctamente).



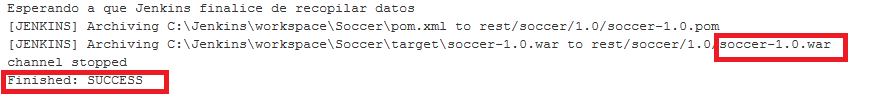
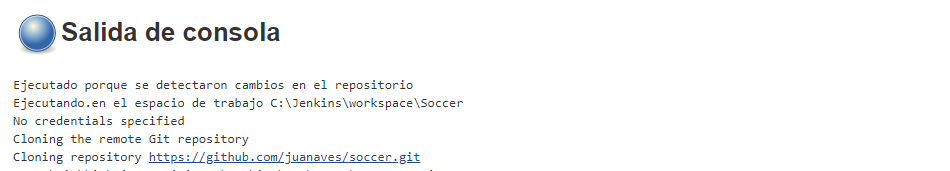
Pulsando sobre dicha construcción, accederemos al **detalle** de la misma. En este caso, el detalle nos indica **“Compilar #1 (07-ene-2018 13:06:10)”**.



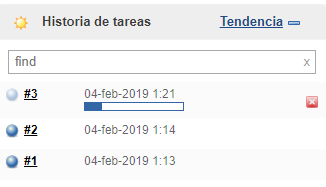
Podemos también ver el log correspondiente. Para ello, pulsaremos sobre **“Console Output”**.



Como vemos, aparecen trazas de git, de maven y, por último, del archivo “.war” **generado**, con un resultado global de éxito (**“Finished: SUCCESS”**).

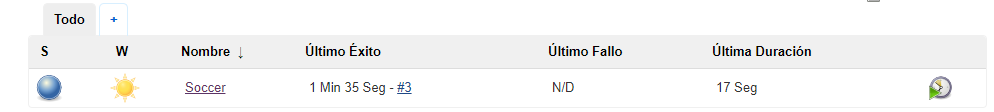


Vamos ahora a volver a nuestro proyecto (pulsando sobre **“Volver al proyecto”**) y vamos a lanzar de nuevo (esta vez de manera manual) la construcción de nuestro proyecto. Para ello, pulsaremos sobre **“Construir ahora”**. 



Si ahora prestamos atención más detenidamente a la sección **“Historia de tareas”**, vemos que ha aparecido un nuevo elemento **“#2”**, que tiene junto a él una **barra de progreso**. Una vez completada la tarea, podremos acceder al detalle de la misma de igual forma que hicimos con **“#1”**. Sobra decir que construcciones sucesivas crearán en esta sección las tareas **“#3”**, **“#4”**, **“#5”**, y así sucesivamente, y que podremos acceder al detalle de las mismas en el momento que consideremos oportuno.

Si ahora volvemos a nuestro panel de control principal (pulsando sobre **“Volver al Panel de Control”**), veremos que, en la línea correspondiente a nuestra tarea **“Soccer”**, aparecerá un icono de un **solecito**  en la segunda columna, lo cual indica que las ejecuciones asociadas han sido **satisfactorias**. La naturaleza de este icono variará dependiendo del resultado de nuestras ejecuciones (habrá un icono relacionado con un **estado atmosférico** peor conforme nuestro ratio de ejecuciones haya sido menos satisfactorio). En nuestro caso, han sido 2 ejecuciones correctas de 2 intentos realizados, de ahí que aparezca un **sol** (de otro modo, aparecerían **nubes**, **tormentas**, etc.).

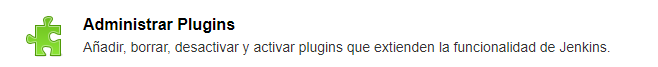


**Configuración de despliegue para Tomcat**

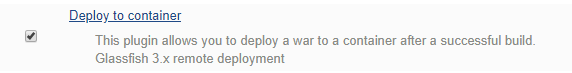
En la sección anterior vimos cómo **compilar** desde **Jenkins** un proyecto **Maven** existente en un repositorio público en **GitHub**. Sin embargo, podemos ir más allá (de hecho, mucho más allá), desplegando, por ejemplo, nuestro proyecto (que se trata de una aplicación web) en un servidor **Tomcat**. Vamos en primer lugar a configurarlo todo.

Para ello, en primer lugar accedemos a la sección de administración de Jenkins (pulsamos sobre **“Administrar Jenkins”**).

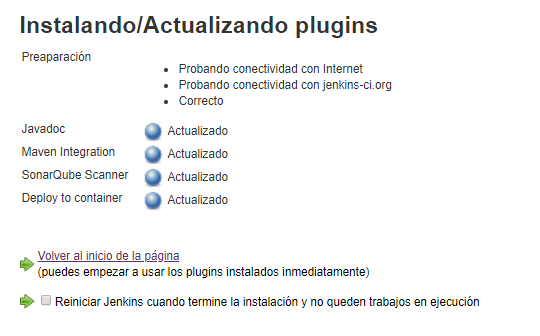
Vamos entonces a la sección de administración de plugins (pulsamos sobre **“Administrar Plugins”**).



Desde esta sección, y de igual modo que antes, vamos a la pestaña **“Todos los plugins”**, y filtramos tecleando **“container”**. Seleccionamos el plugin nombrado como **“Deploy to container ”** y pulsamos sobre **“Instalar sin reiniciar”**.



Dejamos que se instale nuestro plugin. Por seguridad (para asegurarnos de que el plugin se haya instalado correctamente), se recomienda realizar en este punto un **reinicio de jenkins** (“http://localhost:9090/restart”), aunque quizás no sea necesario.



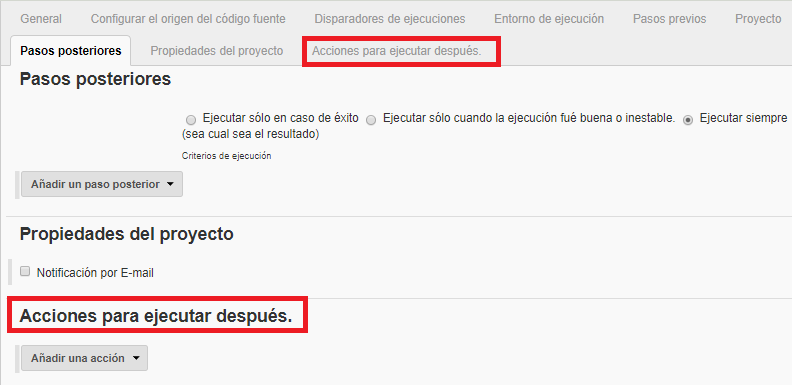
Vamos ahora a configurar nuestra tarea para que, una vez compilado el proyecto, **se despliegue en un servidor Tomcat** que previamente tengamos instalado en nuestra máquina. Para ello, desde nuestra pantalla inicial de bienvenida de Jenkins, **pulsamos sobre la tarea** en cuestión (en mi caso, **“Soccer”**), y nos aparecerá la siguiente pantalla:



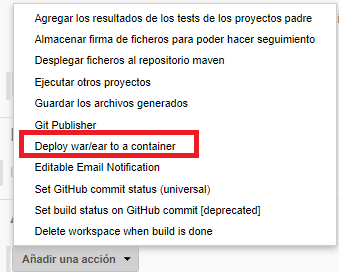
Una vez dentro de la tarea, pulsamos sobre **“Configurar”**. 

Como vemos, esto nos lleva a la **pantalla de configuración** que, cuando creamos la tarea, nos permitió realizar su configuración inicial.

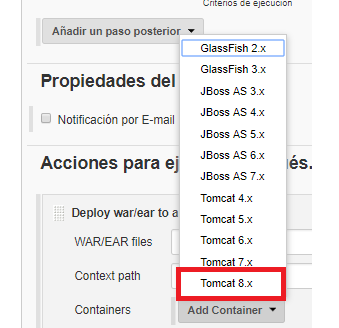
Vamos a la sección “Acciones para ejecutar después”.



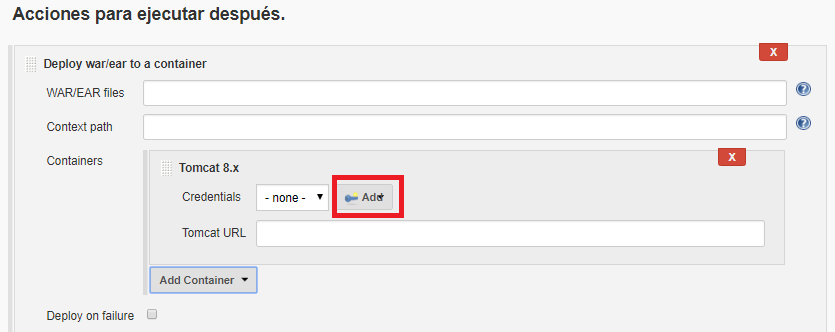
Seleccionamos la opción “**Deploy war/ear to a container**”.



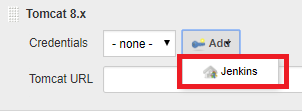
Pulsamos sobre **“Add Container”**, e indicamos a continuación el contenedor sobre el que desplegaremos nuestro “.war” (en mi caso, un servidor **“Tomcat 8.x”**).



Aquí hemos de configurar 2 cosas: las **credenciales de acceso** a nuestro Tomcat, y la **URL** del propio Tomcat.

Para las credenciales, pulsamos sobre **“Ad”**, 

seleccionamos Jenkins (como proveedor de credenciales)



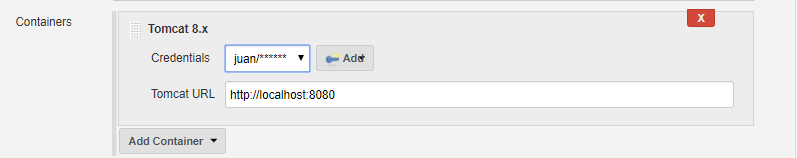
e indicamos el **usuario/password** de nuestro Tomcat.



Este **usuario/password** dependerá de los usuarios que tengamos configurados en nuestro servidor en el archivo **“tomcat-users.xml”** (en mi caso, este archivo lo tengo en la ruta **“C:\Program Files\Apache Software Foundation\Tomcat 8.5\conf\tomcat-users.xml”**).

Pulsamos Add.

Lo siguiente es teclear la **URL de nuestro servidor Tomcat**. En mi caso, lo tengo en el puerto **“8080”**, así que la URL será **“http://localhost:8080”**).



Ya estaría todo. Ahora pulsaremos sobre el botón **“Guardar”** que se encuentra en la parte inferior.

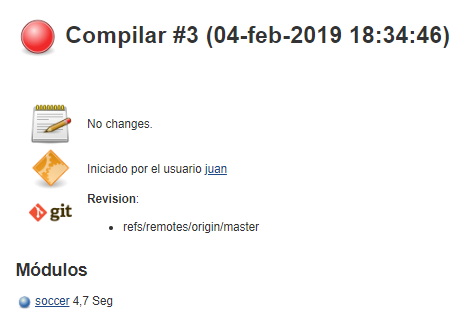
**Despliegue de aplicaciones**

Veamos si todo ha funcionado correctamente. Desde la pantalla principal de nuestra tarea pulsamos sobre **“Construir ahora”**.

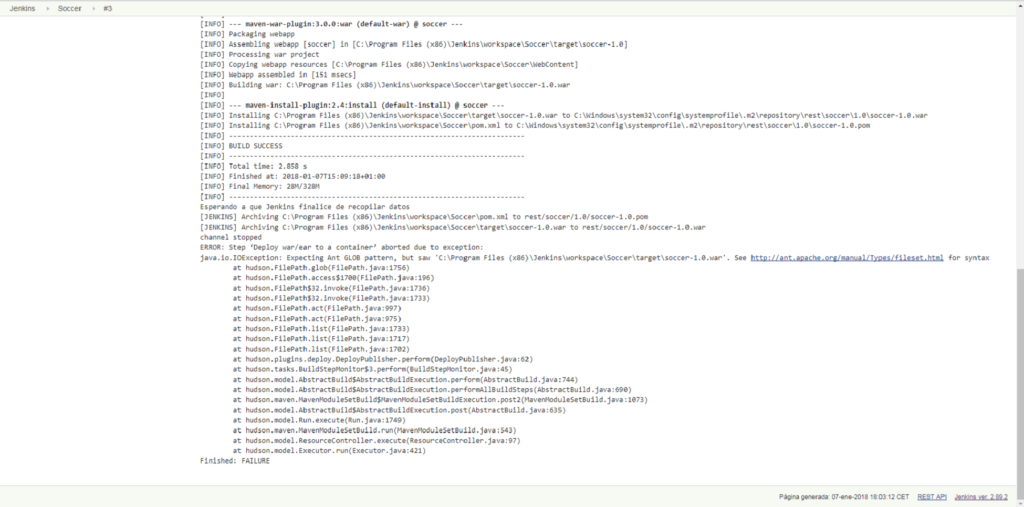
Aparecera una **bolita roja** en el **histórico de tareas**, así que algo ha fallado.



Vamos a investigar lo sucedido. Desde la tarea **“#3”** pulsamos sobre **“Console Output”** para ver el log.



Al final de la traza vemos el error “**java.io.IOException: Expecting Ant GLOB pattern, but saw ‘C:\Program Files (x86)\Jenkins\workspace\Soccer\target\soccer-1.0.war'**”.



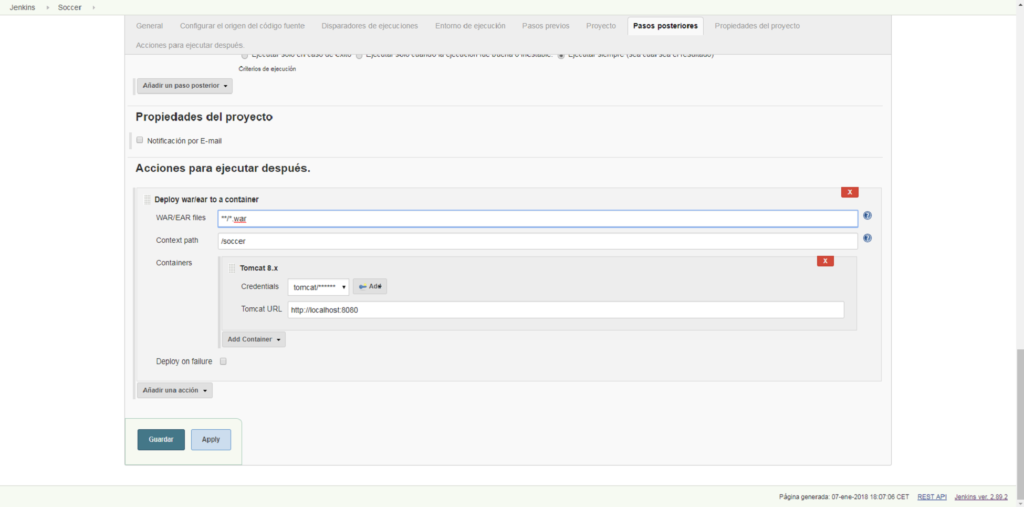
Esto es porque hemos indicado una **ruta específica en lugar de un patrón** a la hora de definir el “.war” a desplegar. Para modificarlo, accedemos a la pantalla de configuración de nuestra tarea y, en la sección **“Acciones para ejecutar después”**, tecleamos **“\*\*/\*.war”** en la propiedad **“WAR/EAR files”**.

Además deberemos **tener un usuario en Tomcat con derechos de implementación**

Para una implementación exitosa de Jenkins Tomcat de un archivo WAR, debe agregar un nuevo usuario a Tomcat con derechos de script de administrador. Puede hacerlo con una edición del archivo tomcat **users.xml** , que se puede encontrar en el directorio conf de Tomcat .

<! - El usuario desplegará el archivo WAR de Jenkins a Tomcat ->  
<user username = "war-deployer" password = "jenkins-tomcat-deploy"  
                                      roles = "manager-script" />

Después de editar el archivo tomcat-users.xml, es una buena idea **reiniciar** el servidor Tomcat para confirmar que los cambios han tenido efecto.

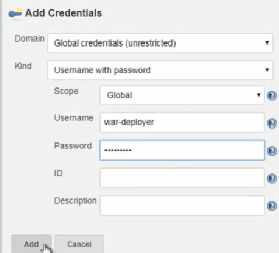


Para configurar las credenciales, debe hacer clic en el botón Agregar junto al campo de entrada vacío y crear un nuevo objeto de credenciales Jenkins:

El nombre de usuario y la contraseña deben coincidir con lo ingresado en el archivo tomcat-users.xml en un paso anterior:

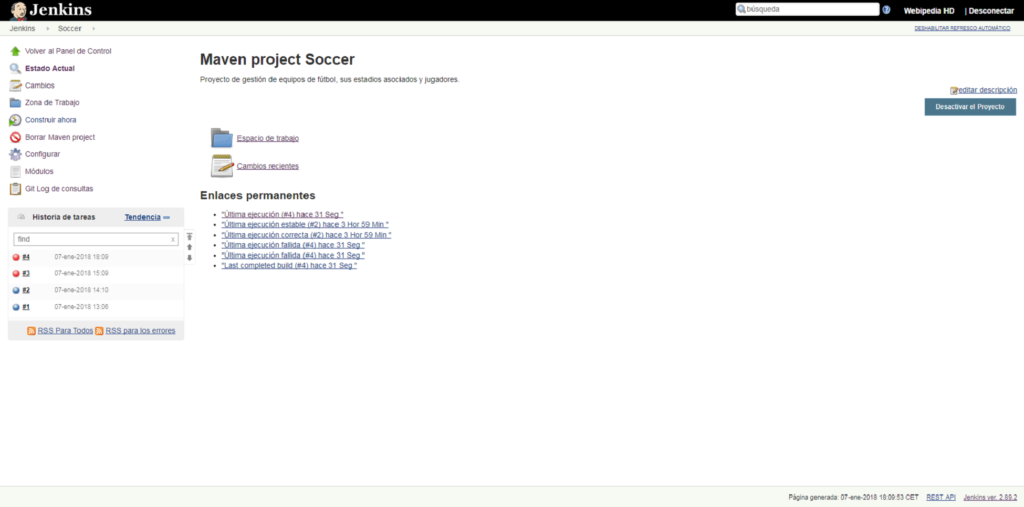
Nombre de usuario: war-deployer

Contraseña: jenkins-tomcat-deploy

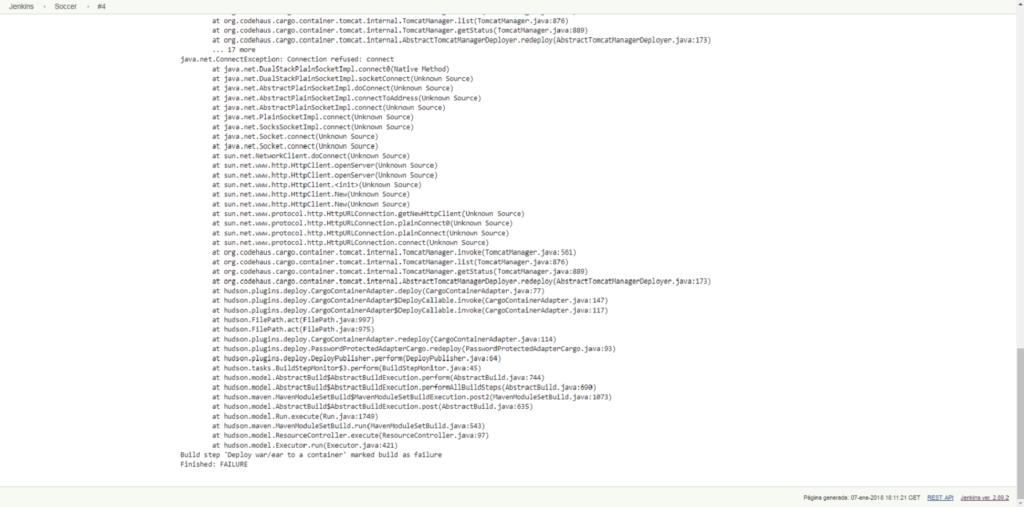


Pulsamos a Añadir y posteriormente guardamos todo.

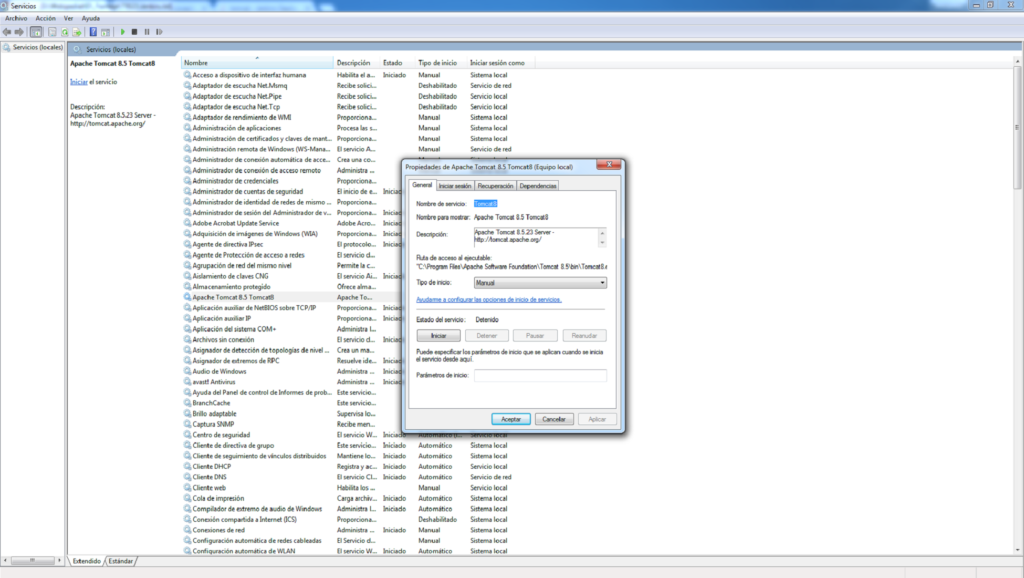
De nuevo, pulsamos sobre **“Construir ahora”**. Vemos que, por desgracia, **ha vuelto a fallar**.



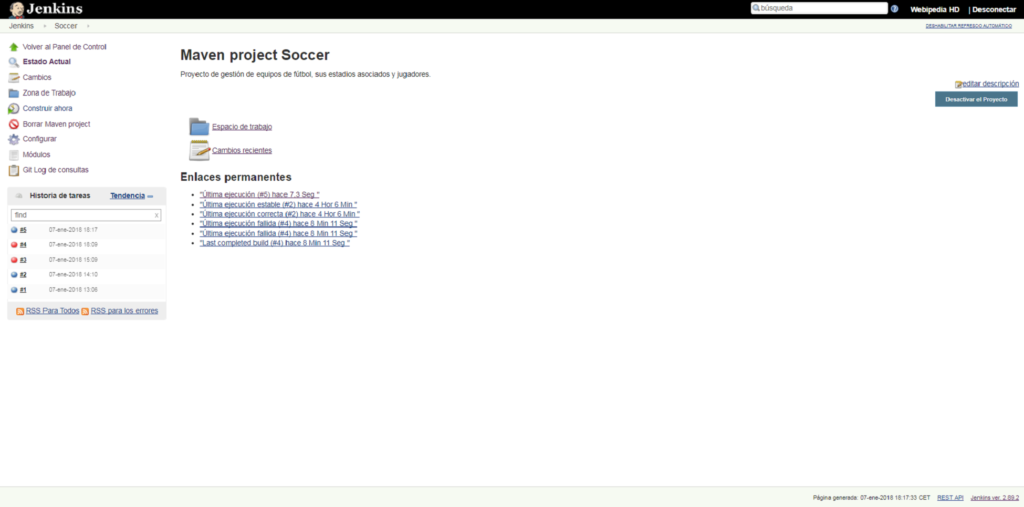
Investiguemos de nuevo el log (**“Console Output”**). La traza nos indica ahora un error **“java.net.ConnectException: Connection refused: connect”**.



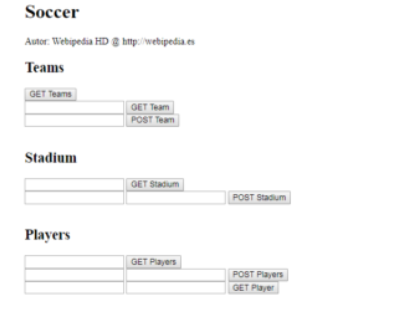
Este error posiblemente se deba a que **no tenemos iniciado el servicio** correspondiente a nuestro servidor **Tomcat**, por lo que Jenkins no es capaz de conectar con él. Desde la **sección de servicios de nuestro PC** vemos que, efectivamente, el servicio **“Apache Tomcat 8.5 Tomcat8”** se encuentra parado. Lo **arrancamos manualmente** (también podemos configurarlo para que se inicie automáticamente al arrancar el PC).



Volvemos de nuevo a nuestro **Jenkins**, e iniciamos una nueva ejecución (pulsamos sobre **“Construir ahora”** de nuevo). Ahora la **bolita** que nos aparece es **azul**, con lo que Jenkins nos indica que **la ejecución se ha llevado a cabo satisfactoriamente**.



Vamos a hacer una prueba accediendo a la URL de nuestra aplicación web, que en mi caso sería **“http://localhost:8080/soccer”**. Y sí, efectivamente, todo ha ido a las 1000 maravillas y **mi aplicación web está en funcionamiento sobre mi servidor Tomcat**.



Y esto sería todo (que no es poco). Cualquier modificación que, a partir de ahora, realice sobre mi repositorio en **GitHub**, tendrá el mismo efecto: **Jenkins**, que se encuentra haciendo **pooling** sobre el repositorio cada minuto, compilaría automáticamente el proyecto **Maven**, generaría el archivo **“.war”**, y lo desplegaría en mi servidor **Tomcat**. Evidentemente, la **velocidad de refresco** se podría relajar para que el pooling se realizara, por ejemplo, cada 5 minutos, aunque eso ya dependerá de las “prisas” de cada uno.

<https://www.theserverside.com/video/Step-by-step-Jenkins-Tomcat-deploy-of-a-WAR-file>