

Despliegue con Jenkins, Tomcat y MariaDB



DESPLIEGUE DE APLICACIONES WEB

SANDRA RUIZ JIMÉNEZ

2º DAW

Lista de contenidos

1. INTRODUCCIÓN	2
2. CREACIÓN DE LOS CONTENEDORES	2
2.1. CREANDO EL DOCKER-COMPOSE	1
3. CONFIGURACIÓN DE TOMCAT	2
4. CONFIGURACIÓN DE LA BASE DE DATOS EN MARIADB	5
5. CONFIGURACIÓN DE LA APLICACIÓN	8
6. SUBIR EL PROYECTO A GITHUB	10
7. CONFIGURACIÓN BÁSICA DE JENKINS	11
8. CONFIGURACIÓN ESPECÍFICA DE JENKINS	14
9. DESPLIEGUE DE LA APLICACIÓN	16
BIBLIOGRAFÍA	24

1. INTRODUCCIÓN

El objetivo de esta memoria es crear los contenedores necesarios y realizar la correspondiente configuración para que un proyecto de Jenkins compile un proyecto CRUD alojado en GitHub y lo despliegue en Tomcat.

2. CREACIÓN DE LOS CONTENEDORES

Si nuestra aplicación es un proyecto CRUD que necesita tener acceso a BBDD, quiere decir que además de los contenedores de Tomcat y Jenkins, necesitamos otro para la base de datos. Para que todos los contenedores puedan interactuar entre ellos, es preciso que se encuentren en la misma red. Por este motivo, vamos a crear los contenedores a través de un fichero docker-compose, que automáticamente los añade a la misma red y así no tendremos dificultades.

2.1. CREANDO EL DOCKER-COMPOSE

A continuación se muestra una captura de nuestro archivo docker-compose.yml:

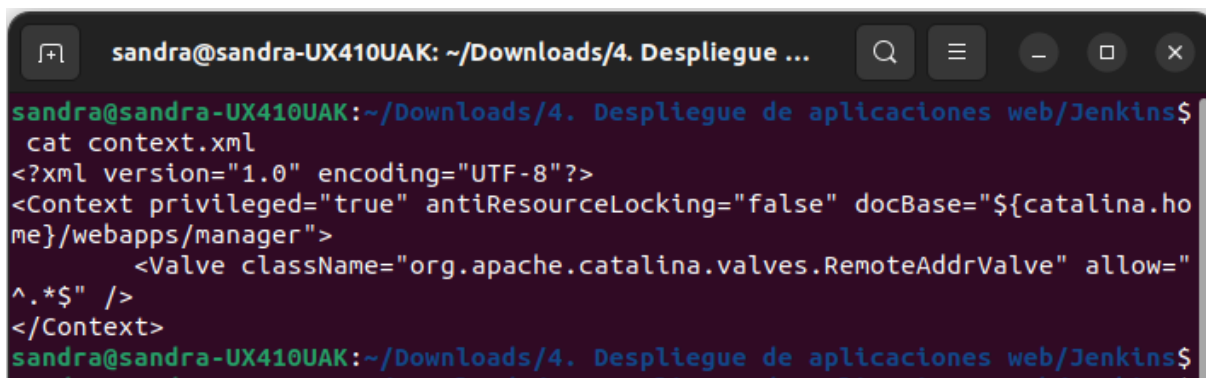
```
1  version: "3.7" #versión a actualizar
2
3  services:
4
5      tomcat: #servicio que se va a desplegar al ejecutar el compose
6          container_name: jenkinsTomcat #nombre que le voy a asignar al contenedor que creará el compose
7          image: tomcat:10.0 #imagen de tomcat que voy a usar para el contenedor
8          ports:
9              - 8888:8080 #mapeo de puertos, del 8080 del contenedor al 8887 de mi localhost
10         volumes:
11             - ./context.xml:/usr/local/tomcat/conf/context.xml #archivo context.xml configurado
12             - ./tomcat-users.xml:/usr/local/tomcat/conf/tomcat-users.xml #archivo tomcat-users.xml configurado
13
14     mariadb: #servicio que se va a desplegar al ejecutar el compose
15         container_name: jenkinsDB
16         image: mariadb #imagen de mariaDB que vamos a utilizar
17         ports:
18             - 3306:3306
19         environment: #Se establecen las distintas variables de entorno que se usarán para configurar el servicio
20             MYSQL_ROOT_PASSWORD: '1234' #contraseña de la base de datos
21         volumes: #declaramos la ruta del volumen que se va a montar
22             - '/db:/var/lib/mariadb'
23
24     jenkins: #servicio que se va a desplegar al ejecutar el compose
25         container_name: jenkinsCompose
26         image: jenkins/jenkins:ls #imagen de jenkins que voy a utilizar
27         ports:
28             - 8091:8080
29             - 50000:50000
30
```

IMPORTANTE: es necesario usar la versión 10.0 de Tomcat, porque la 9.0 da errores al desplegar aplicaciones de Spring.

En este fichero, declaramos los tres servicios que vamos a necesitar, Tomcat, MariaDB y Jenkins.

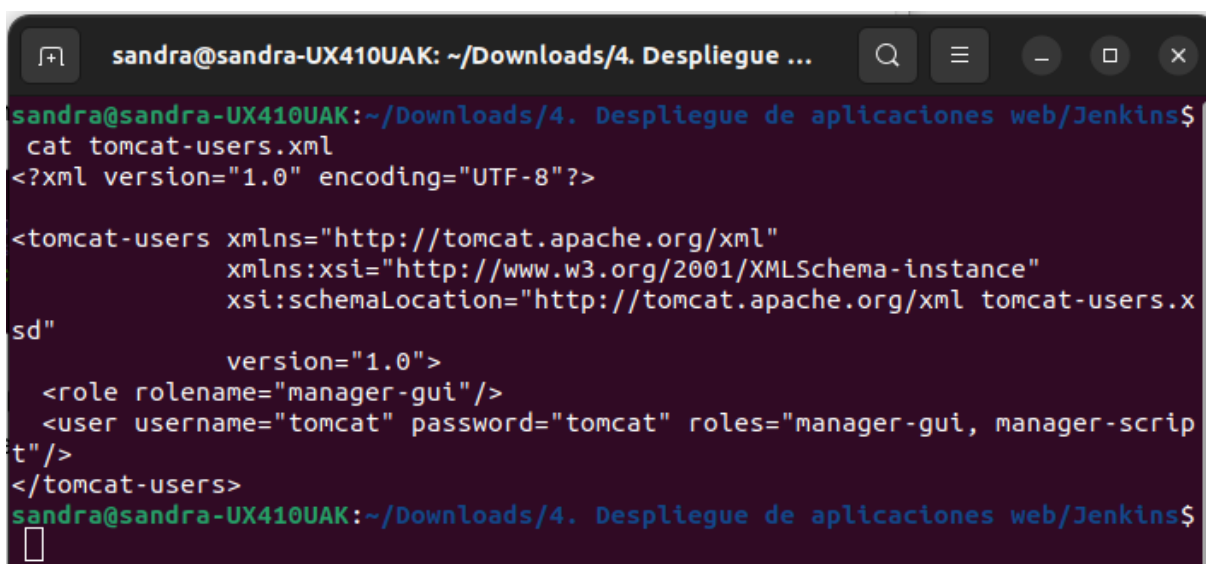
Para el servicio de tomcat necesitamos incorporar en el mismo directorio del docker-compose.yml los ficheros **context.xml** y **tomcat-users.xml**, los cuales están configurados para tener acceso al manager de Tomcat. A continuación veremos sus cambios respecto a los ficheros originales.

El fichero **context.xml** queda así:



```
sandra@sandra-UX410UAK: ~/Downloads/4. Despliegue de aplicaciones web/Jenkins$ cat context.xml
<?xml version="1.0" encoding="UTF-8"?>
<Context privileged="true" antiResourceLocking="false" docBase="${catalina.home}/webapps/manager">
    <Valve className="org.apache.catalina.valves.RemoteAddrValve" allow="^.*$" />
</Context>
```

Y el fichero **tomcat-users.xml**, así:



```
sandra@sandra-UX410UAK: ~/Downloads/4. Despliegue de aplicaciones web/Jenkins$ cat tomcat-users.xml
<?xml version="1.0" encoding="UTF-8"?>

<tomcat-users xmlns="http://tomcat.apache.org/xml"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
               version="1.0">
    <role rolename="manager-gui"/>
    <user username="tomcat" password="tomcat" roles="manager-gui, manager-script"/>
</tomcat-users>
```

Como vemos hemos añadido el rol de manager-gui y el de manager-script.

Abrimos una terminal desde el directorio que contiene el compose y ejecutamos el comando **docker-compose up**

```
sandra@sandra-UX410UAK: ~/Downloads/4. Despliegue d...
sandra@sandra-UX410UAK:~/Downloads/4. Despliegue de aplicaciones web/Jenkins$
docker-compose up
Creating jenkinsDB      ... done
Creating jenkinsTomcat  ... done
Creating jenkinsCompose ... done

Attaching to jenkinsCompose, jenkinsDB, jenkinsTomcat
jenkinsDB | 2023-02-14 09:28:08+00:00 [Note] [Entrypoint]: Entrypoint script
for MariaDB Server 1:10.10.2+maria~ubu2204 started.
jenkinsTomcat | NOTE: Picked up JDK_JAVA_OPTIONS:  --add-opens=java.base/java.
lang=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.ba
se/java.util=ALL-UNNAMED --add-opens=java.base/java.util.concurrent=ALL-UNNAME
D --add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED
jenkinsDB | 2023-02-14 09:28:09+00:00 [Note] [Entrypoint]: Switching to dedic
ated user 'mysql'
jenkinsDB | 2023-02-14 09:28:09+00:00 [Note] [Entrypoint]: Entrypoint script
for MariaDB Server 1:10.10.2+maria~ubu2204 started.
jenkinsDB | 2023-02-14 09:28:09+00:00 [Note] [Entrypoint]: Initializing datab
ase files
jenkinsCompose | Running from: /usr/share/jenkins/jenkins.war
jenkinsCompose | webroot: EnvVars.masterEnvVars.get("JENKINS_HOME")
jenkinsTomcat | 14-Feb-2023 09:28:10.016 INFO [main] org.apache.catalina.start
```

En otro terminal, con el comando **docker ps**, comprobamos que ambos contenedores se han creado:

```
sandra@sandra-UX410UAK: ~
sandra@sandra-UX410UAK:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
2d29e4876e55   tomcat:9.0     "catalina.sh run"        31 seconds ago Up 29 seconds 0.0.0.
0:8888->8080/tcp, :::8888->8080/tcp
jenkinsTomcat
baa181a0e47c   jenkins/jenkins:lts "/usr/bin/tini -- /u..." 31 seconds ago Up 29 seconds 0.0.0.
0:50000->50000/tcp, :::50000->50000/tcp, 0.0.0.0:8091->8080/tcp, :::8091->8080/tcp
jenkinsCompose
9ccd7d45cbb3   mariadb        "docker-entrypoint.s..." 31 seconds ago Up 29 seconds 0.0.0.
0:3306->3306/tcp, :::3306->3306/tcp
jenkinsDB
b1bfa24f586a   phpmyadmin/phpmyadmin "/docker-entrypoint...." 3 months ago   Up 33 minutes 0.0.0.
0:8099->80/tcp, :::8099->80/tcp
phpmyadminDawes
sandra@sandra-UX410UAK:~$
```

3. CONFIGURACIÓN DE TOMCAT

Para que nos aparezca la página de Tomcat debemos comprobar que el directorio webapps (directorio que contiene las aplicaciones web) contiene el host-manager, el manager y el directorio docs.

Sin embargo, estos ficheros se encuentran dentro de webapps.dist por lo tenemos que copiarlos en el directorio webapps.

Entramos dentro de su sistema de ficheros:

```
docker exec -it jenkinsTomcat /bin/bash
```

A continuación comprobamos que están dentro del directorio webapps.dist

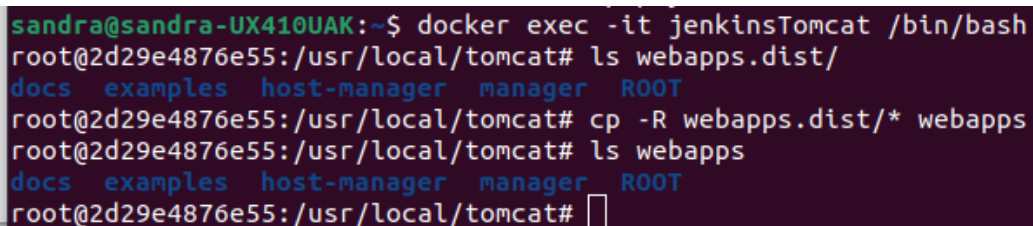
```
ls webapps.dist/
```

Como están, para copiarlos, ejecutamos el siguiente comando:

```
cp -R webapps.dist/* webapps
```

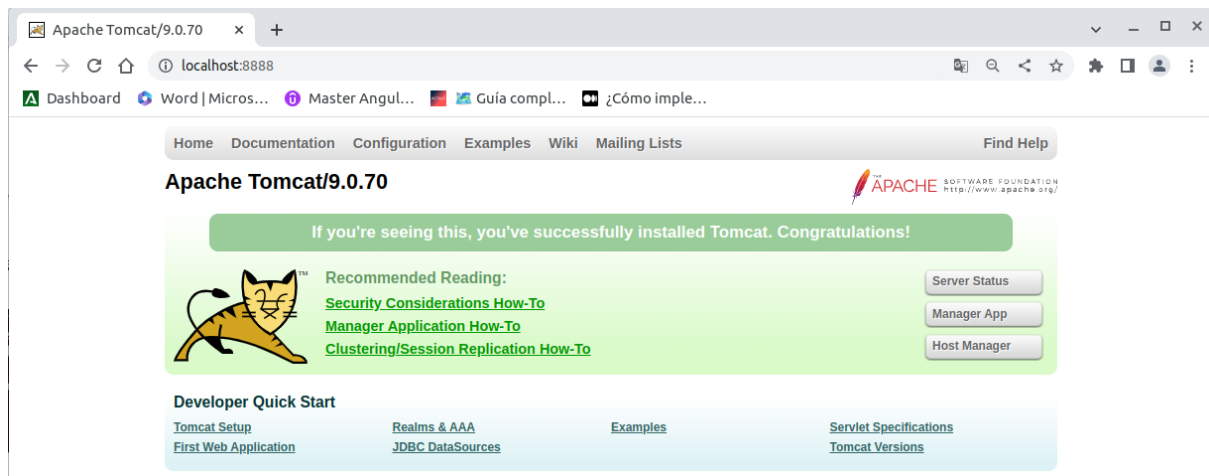
Comprobamos que se han copiado dentro del directorio webapps:

```
ls webapps
```



```
sandra@sandra-UX410UAK:~$ docker exec -it jenkinsTomcat /bin/bash
root@2d29e4876e55:/usr/local/tomcat# ls webapps.dist/
docs  examples  host-manager  manager  ROOT
root@2d29e4876e55:/usr/local/tomcat# cp -R webapps.dist/* webapps
root@2d29e4876e55:/usr/local/tomcat# ls webapps
docs  examples  host-manager  manager  ROOT
root@2d29e4876e55:/usr/local/tomcat#
```

Después de esto, nos debería aparecer la página de inicio de Tomcat en el navegador, escribiendo localhost:8888:



Sin embargo, aunque hemos configurado el manager para entrar y tener permiso, aún no nos lo permite. Esto se debe a que Tomcat no está utilizando el fichero context.xml que hemos modificado, sino el fichero context.xml que se encuentra en la ruta /usr/local/tomcat/webapps/manager/META-INF.

Para solucionarlo, tan solo tenemos que borrar este archivo context.xml:

rm context.xml

```
root@2d29e4876e55:/usr/local/tomcat# cd webapps/manager/META-INF
root@2d29e4876e55:/usr/local/tomcat/webapps/manager/META-INF# rm context.xml
root@2d29e4876e55:/usr/local/tomcat/webapps/manager/META-INF#
```

Ahora reiniciamos Tomcat para que se apliquen los cambios. Para reiniciar, debemos ir al directorio /usr/local/tomcat/bin y ejecutar ./startup.sh.

```
root@2d29e4876e55:/usr/local/tomcat# cd bin/
root@2d29e4876e55:/usr/local/tomcat/bin# ./startup.sh
Using CATALINA_BASE:   /usr/local/tomcat
Using CATALINA_HOME:   /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:        /opt/java/openjdk
Using CLASSPATH:        /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
root@2d29e4876e55:/usr/local/tomcat/bin#
```

Si ahora intentamos entrar al manager de nuevo, usando usuario y contraseña tomcat, ya debemos tener acceso al manager.

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

4. CONFIGURACIÓN DE LA BASE DE DATOS EN MARIADB

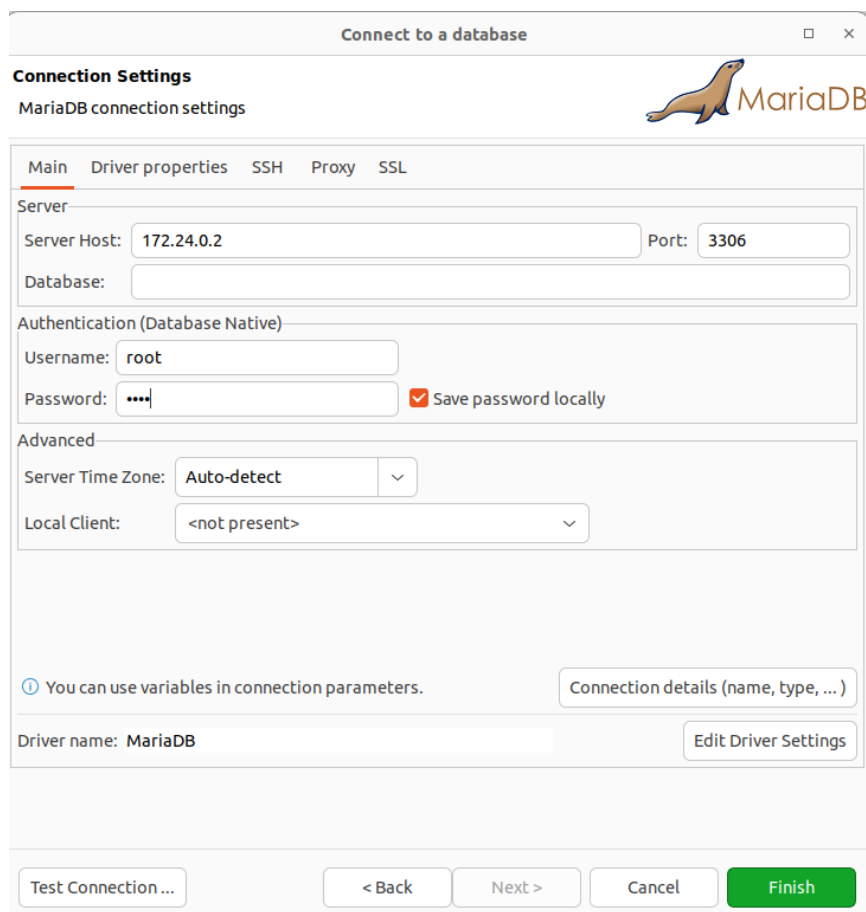
Usando el DBeaver, creamos una nueva base de datos de MariaDB.

Primero necesitamos conocer la IP del contenedor de MariaDB que hemos creado. Para ello, usamos el comando `docker inspect [nombre-contenedor]`


```
sandra@sandra-UX410UAK: ~  
    "IPAMConfig": null,  
    "Links": null,  
    "Aliases": [  
        "mariadb",  
        "9ccd7d45cbb3"  
    ],  
    "NetworkID": "c4041c5684f92713f9da9af91441cbbd37d9f868  
a2b9cccd2c0f735c5d08300f",  
    "EndpointID": "72020e5b73a59b795663741f1e9175d63bb9a73  
f2c79b563148b7152d25e4bee",  
    "Gateway": "172.24.0.1",  
    "IPAddress": "172.24.0.2",  
    "IPPrefixLen": 16,  
    "IPv6Gateway": "",  
    "GlobalIPv6Address": "",  
    "GlobalIPv6PrefixLen": 0,  
    "MacAddress": "02:42:ac:18:00:02",  
    "DriverOpts": null  
    }  
  }  
}  
]  
sandra@sandra-UX410UAK:~$
```

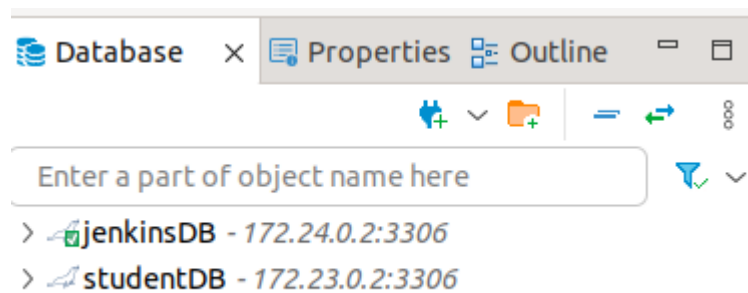
Nuestra IP es la **172.24.0.2**.

Creamos la conexión en el DBeaver:



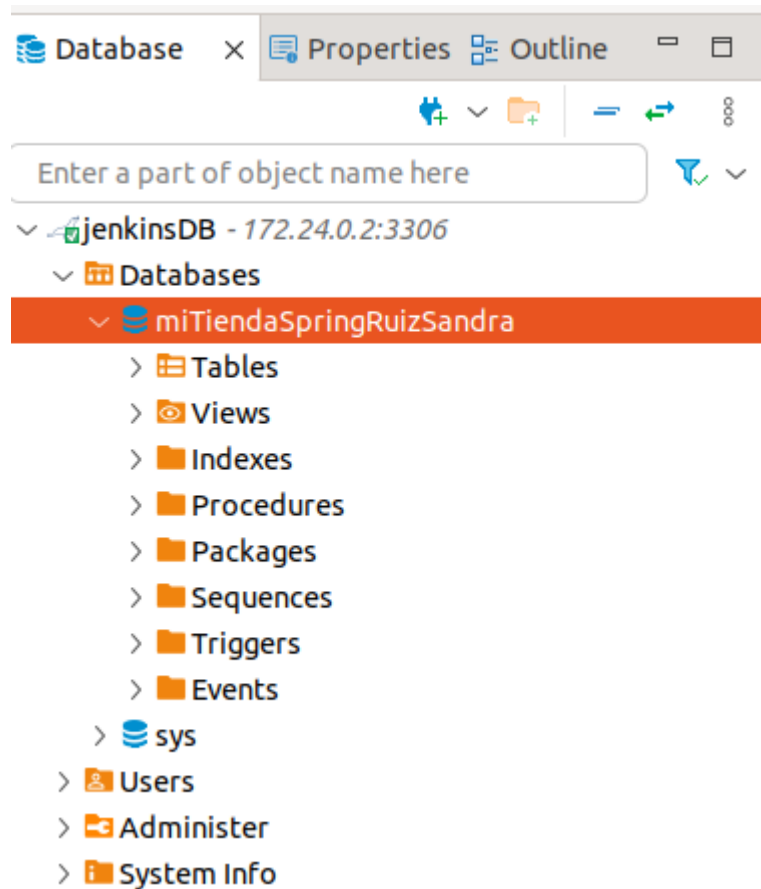
The screenshot shows the 'Connect to a database' dialog box in DBeaver, specifically the 'Main' tab for a MariaDB connection. The window title is 'Connect to a database'. Below the title bar, it says 'Connection Settings' and 'MariaDB connection settings'. The MariaDB logo is visible on the right. The 'Main' tab is selected, showing fields for 'Server Host' (172.24.0.2), 'Port' (3306), and 'Database' (empty). The 'Authentication (Database Native)' section has 'Username' set to 'root' and 'Password' masked with dots, with a checked box for 'Save password locally'. The 'Advanced' section shows 'Server Time Zone' set to 'Auto-detect' and 'Local Client' set to '<not present>'. At the bottom, there is a 'Driver name' field with 'MariaDB' and an 'Edit Driver Settings' button. The footer contains buttons for 'Test Connection ...', '< Back', 'Next >', 'Cancel', and a green 'Finish' button.

En Password ponemos la misma contraseña que hemos indicado en el docker-compose, en este caso, 1234.



IMPORTANTE: al ser un proyecto con base de datos, antes de ejecutarlo, el contenedor de la base de datos debe estar levantado y que ésta debe contener un esquema con el nombre que se indicó en el application.properties. De no ser así, al hacer el Run As, la aplicación fallará.

Para ello, creamos una base de datos con el mismo nombre que hemos indicado en el application.properties de nuestro proyecto de Spring y le añadimos los datos.



5. CONFIGURACIÓN DE LA APLICACIÓN

Vamos a desplegar una aplicación de Spring con acceso a base de datos.

Debemos tener en cuenta algunos aspectos previos para que nuestra aplicación funcione en su despliegue:

1. El fichero Application (que encontramos en src/main/java) de nuestra aplicación debe extender de `SpringServletInitializer`.

```
*MiTiendaSpringRuizSandraApplication.java x
1 package com.jacaranda.tienda;
2
3 import java.util.HashMap;
12
13 @SpringBootApplication
14 public class MiTiendaSpringRuizSandraApplication extends SpringServletInitializer {
15
16     public static void main(String[] args) {
17         SpringApplication.run(MiTiendaSpringRuizSandraApplication.class, args);
18     }
19
```

2. En el `application.properties` (en src/main/resources) debemos hacer algunas configuraciones tales como indicar la dirección IP del contenedor de MariaDB que hemos creado previamente. Esto es importante para que nuestra aplicación desde el contenedor de Tomcat se pueda comunicar con la base de datos.

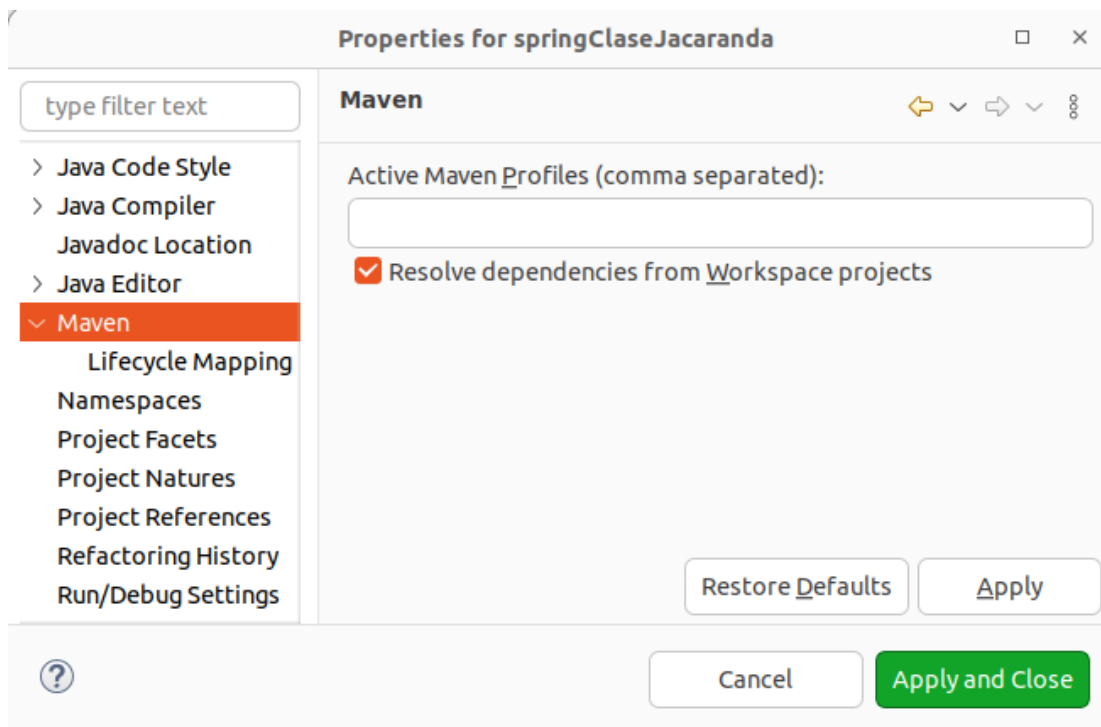
```
application.properties x
1 spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
2 spring.datasource.url=jdbc:mariadb://172.24.0.2:3306/miTiendaSpringRuizSandra?useSSL=false
3 spring.datasource.username=root
4 spring.datasource.password=1234
5 spring.security.user.name=sandra
6 spring.security.user.password=sandra
7 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDB103Dialect
```

Es importante declarar el dialecto que va a usar (línea 7).

3. Añadimos la dependencia de mariadb en el fichero pom.xml

```
41 <dependency>
42     <groupId>org.mariadb.jdbc</groupId>
43     <artifactId>mariadb-java-client</artifactId>
44     <version>2.5.2</version>
45     <scope>runtime</scope>
46 </dependency>
```

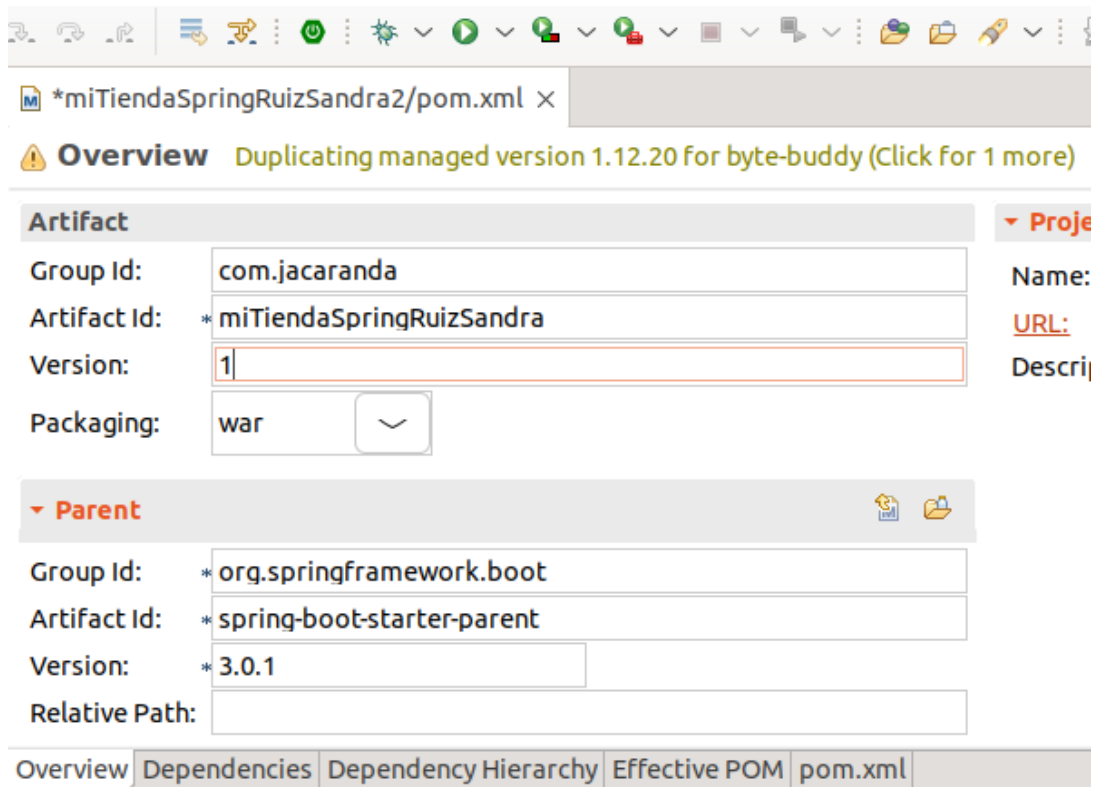
4. En la configuración de Maven, tendremos que dejar la configuración del perfil activo en blanco, la cual por defecto nos especifica que es pom.xml. Así que lo borramos.



5. Comprobamos que las rutas de las vistas de nuestra aplicación están correctamente nombradas. Es decir, revisamos las etiquetas href de los html, así como los href que introducen las hojas de estilo.

Una vez realizados los pasos anteriores, ya solo nos queda conseguir nuestro fichero .war que es el que contendrá a nuestra aplicación.

Para conseguirlo, primero vamos al fichero de configuración pom.xml del proyecto y en la pestaña Overview cambiamos el packaging por defecto jar por war:



Guardamos los cambios y ejecutamos nuestra aplicación a través de Run As → Maven Install.

Una vez terminado, obtendremos un mensaje de éxito si todo ha ido correctamente:

```

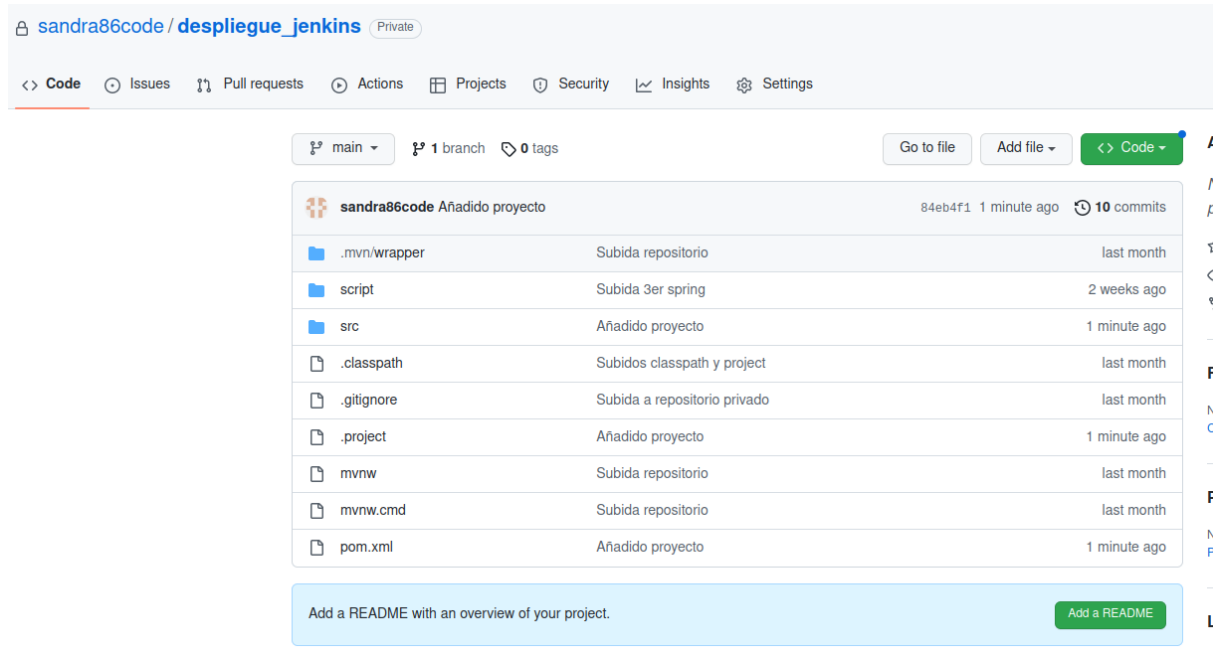
<terminated> /home/sandra/Downloads/eclipse-jee-2022-03/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.2.v20220201-1208/jre/bin/java (F
[INFO]
[INFO] --- spring-boot-maven-plugin:3.0.1:repackage (repackage) @ miTiendaSpringRuizSandra ---
[INFO] Replacing main artifact with repackaged archive
[INFO]
[INFO] --- maven-install-plugin:3.0.1:install (default-install) @ miTiendaSpringRuizSandra ---
[INFO] Installing /home/sandra/entornoservidorespring-workspace/miTiendaSpringRuizSandra2/pom.xml to /ho
[INFO] Installing /home/sandra/entornoservidorespring-workspace/miTiendaSpringRuizSandra2/target/miTiend
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 02:25 min
[INFO] Finished at: 2023-02-14T11:07:03+01:00
[INFO] -----

```

Nuestro archivo .war ya se ha generado y podemos encontrarlo en el directorio /target del proyecto:

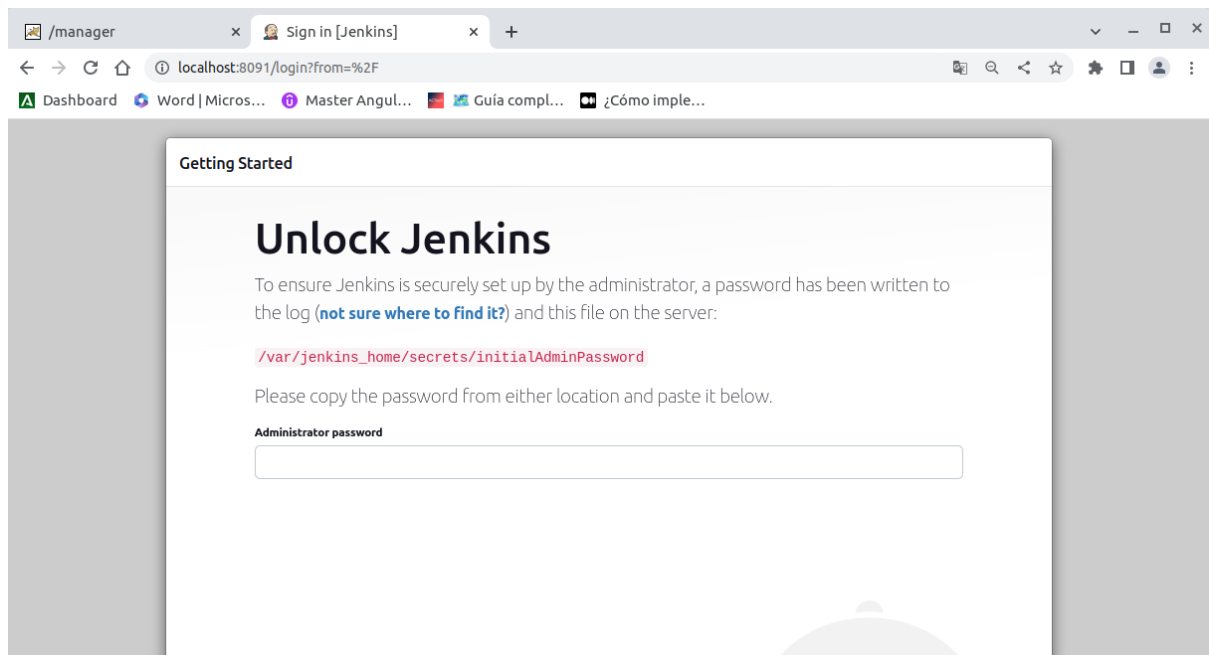
6. SUBIR EL PROYECTO A GITHUB

Subimos nuestro proyecto a un repositorio, en este caso privado, de GitHub:



7. CONFIGURACIÓN BÁSICA DE JENKINS

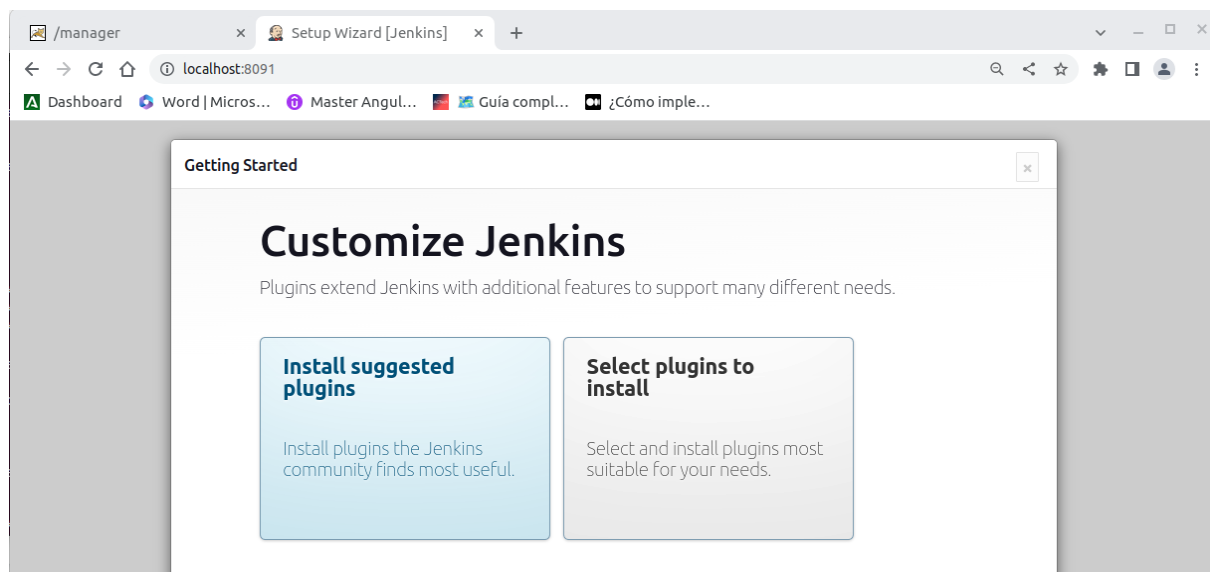
Lo primero que debemos hacer es configurar el usuario de Jenkins. Cuando entramos al navegador por primera vez nos pedirá una clave.



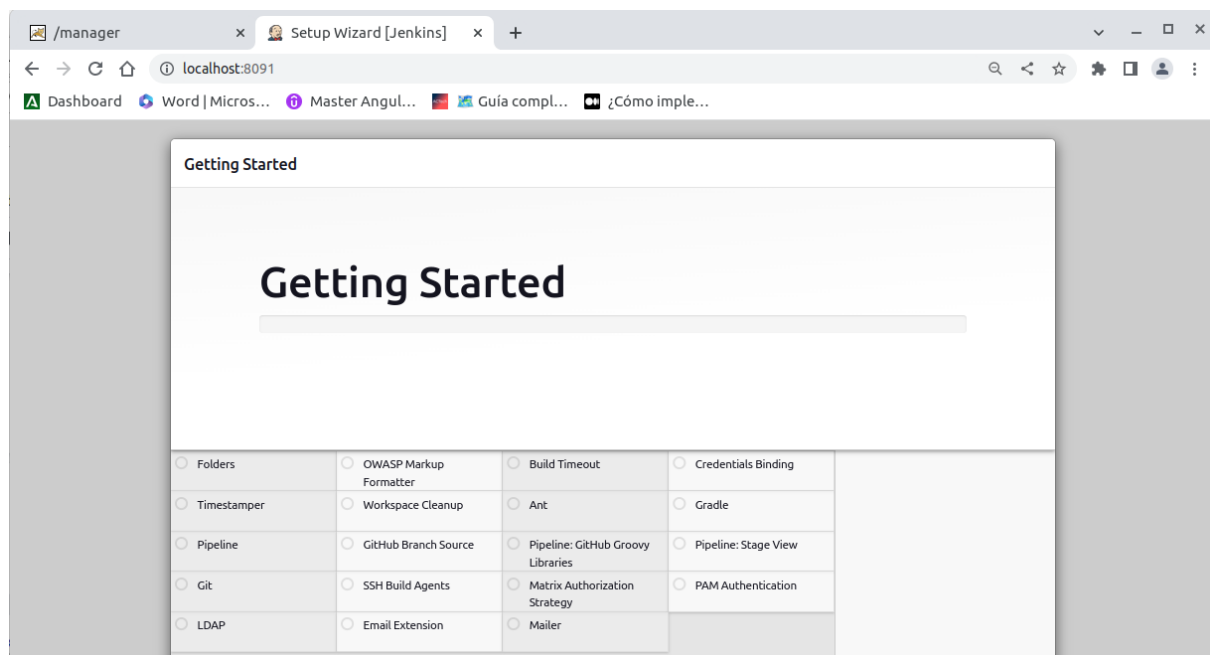
Dicha clave se genera en la instalación del contenedor, nos la muestra en la consola al lanzar el docker-compose, pero también podemos recuperarla en `/var/jenkins_home/secrets/initialAdminPassword`

```
sandra@sandra-UX410UAK: ~/Downloads/4. Despliegue de a...
constructor java.lang.invoke.MethodHandles$Lookup(java.lang.Class,int)
jenkinsCompose | WARNING: Please consider reporting this to the maintainers of o
rg.codehaus.groovy.vmplugin.v7.Java7$1
jenkinsCompose | WARNING: Use --illegal-access=warn to enable warnings of furthe
r illegal reflective access operations
jenkinsCompose | WARNING: All illegal access operations will be denied in a futu
re release
jenkinsCompose | 2023-02-14 10:56:57.003+0000 [id=35] INFO jenkins.install.
SetupWizard#init:
jenkinsCompose | *****
jenkinsCompose | *****
jenkinsCompose | *****
jenkinsCompose | Jenkins initial setup is required. An admin user has been creat
ed and a password generated.
jenkinsCompose | Please use the following password to proceed to installation:
jenkinsCompose | 873fd2ca5b5f4151adfcdf358ca6ca0
jenkinsCompose | This may also be found at: /var/jenkins_home/secrets/initialAdm
inPassword
jenkinsCompose | *****
```

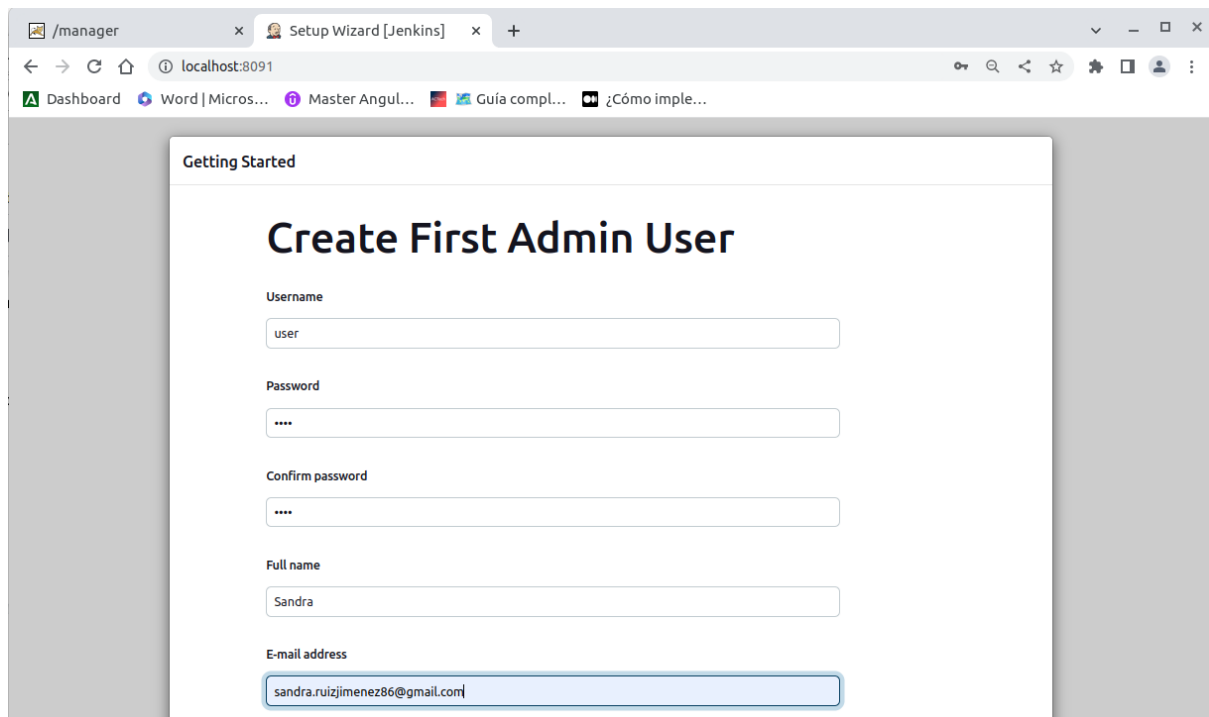
A continuación Jenkins nos pedirá que plugins queremos instalar.



Escogemos la opción de plugins por defecto y comienza su proceso de instalación.



Al terminar nos pide los datos de la cuenta de administrador:

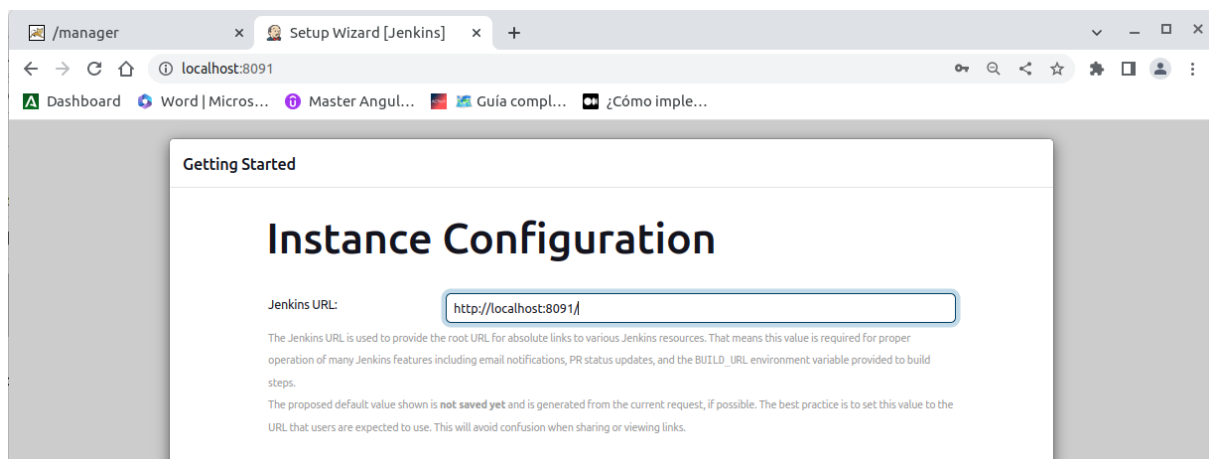


The screenshot shows a web browser window with the address bar at `localhost:8091`. The page title is "Getting Started" and the main heading is "Create First Admin User". The form contains the following fields:

- Username:** `user`
- Password:** `****`
- Confirm password:** `****`
- Full name:** `Sandra`
- E-mail address:** `sandra.ruizjimenez86@gmail.com`

Nuestro nombre de usuario y contraseña serán “user”.

El último paso es configurar la ruta por defecto de Jenkins. Vamos a dejar la que aparece por defecto:

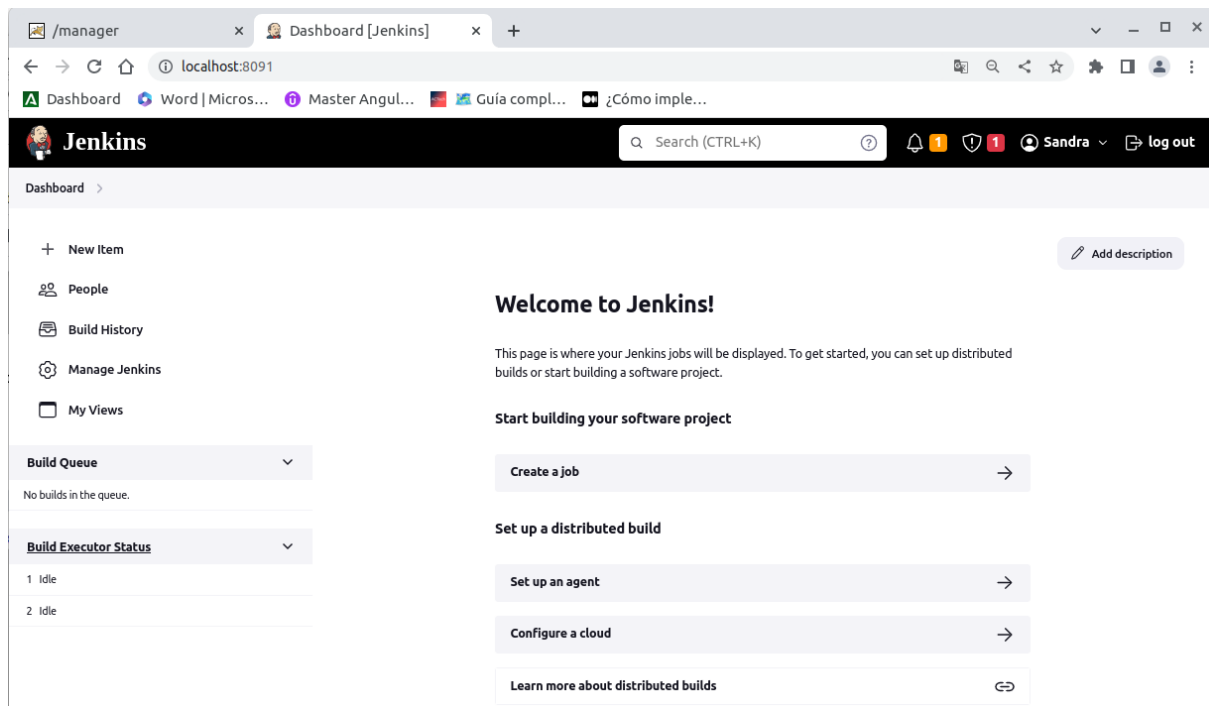


The screenshot shows the "Instance Configuration" page of the Jenkins Setup Wizard. The "Jenkins URL" field is highlighted with a blue border and contains the value `http://localhost:8091/`. Below the field, there is explanatory text:

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 ya tendríamos completa la configuración inicial de Jenkins.



8. CONFIGURACIÓN ESPECÍFICA DE JENKINS

Para que Jenkins pueda recoger los archivos del proyecto que se encuentra alojado en GitHub, lo construya y lo acabe desplegando en Tomcat necesitamos hacer unos pequeños ajustes adicionales.

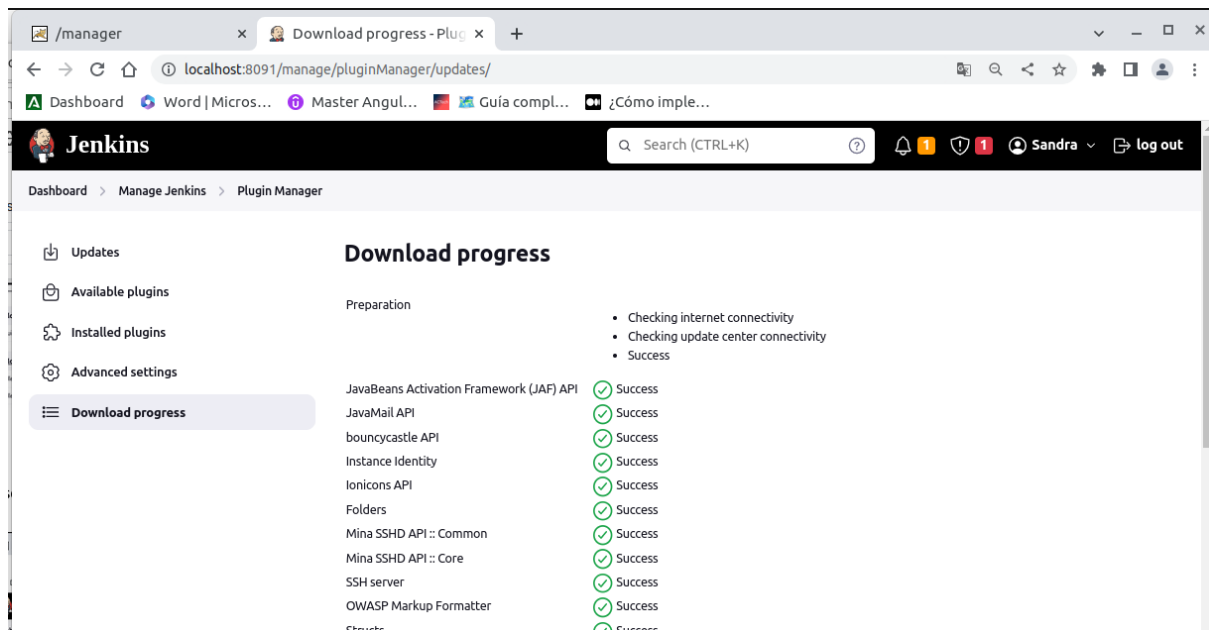
Primero debemos instalar los plugins Maven Integration y Deploy to Container. Para ello, en el panel de control entramos en Manage Jenkins → Manage Plugins

The screenshot shows the Jenkins 'Manage Jenkins' page. The left sidebar contains navigation links: 'New Item', 'People', 'Build History', 'Manage Jenkins' (selected), and 'My Views'. Below these are expandable sections for 'Build Queue' (showing 'No builds in the queue') and 'Build Executor Status' (showing two idle executors). The main content area is titled 'Manage Jenkins' and features a blue notification banner about a new version (2.375.3) and a yellow warning banner about building on the built-in node. The 'System Configuration' section includes four cards: 'Configure System', 'Global Tool Configuration', 'Manage Plugins', and 'Manage Nodes and Clouds'.

Buscamos los plugins que hemos indicado anteriormente y los instalamos:

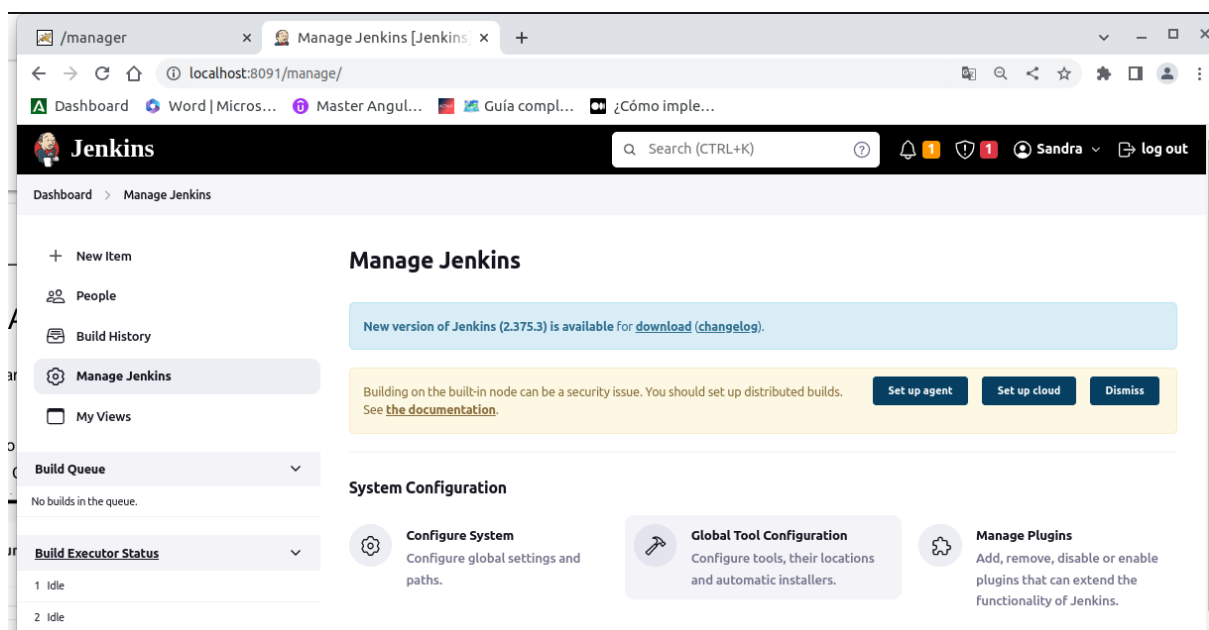
The screenshot shows the Jenkins 'Plugin Manager' page. The left sidebar contains navigation links: 'Updates', 'Available plugins' (selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main content area is titled 'Plugins' and features a search bar with the text 'deploy'. Below the search bar, there's a table listing available plugins. The table has columns for 'Install', 'Name', and 'Released'. Two plugins are listed: 'Maven Integration' (version 3.20) and 'Deploy to container' (version 1.16).

Install	Name	Released
<input checked="" type="checkbox"/>	Maven Integration 3.20 Build Tools This plugin provides a deep integration between Jenkins and Maven. It adds support for automatic triggers between projects depending on SNAPSHOTS as well as the automated configuration of various Jenkins publishers such as Junit.	4 mo 3 days ago
<input checked="" type="checkbox"/>	Deploy to container 1.16 Artifact Uploaders This plugin allows you to deploy a war to a container after a successful build. Glassfish 3.x remote deployment	2 yr 3 mo ago



9. DESPLIEGUE DE LA APLICACIÓN

En este caso nosotros queremos desplegar un proyecto Maven, procedemos a configurarlo. Antes de empezar a configurar el proyecto nuevo, debemos instalar Maven en Jenkins. Para ello, volvemos al Dashboard, seleccionamos de nuevo Manage Jenkins y luego Global Tool Configuration.



Buscamos Maven y le decimos a Jenkins que lo instala automáticamente.

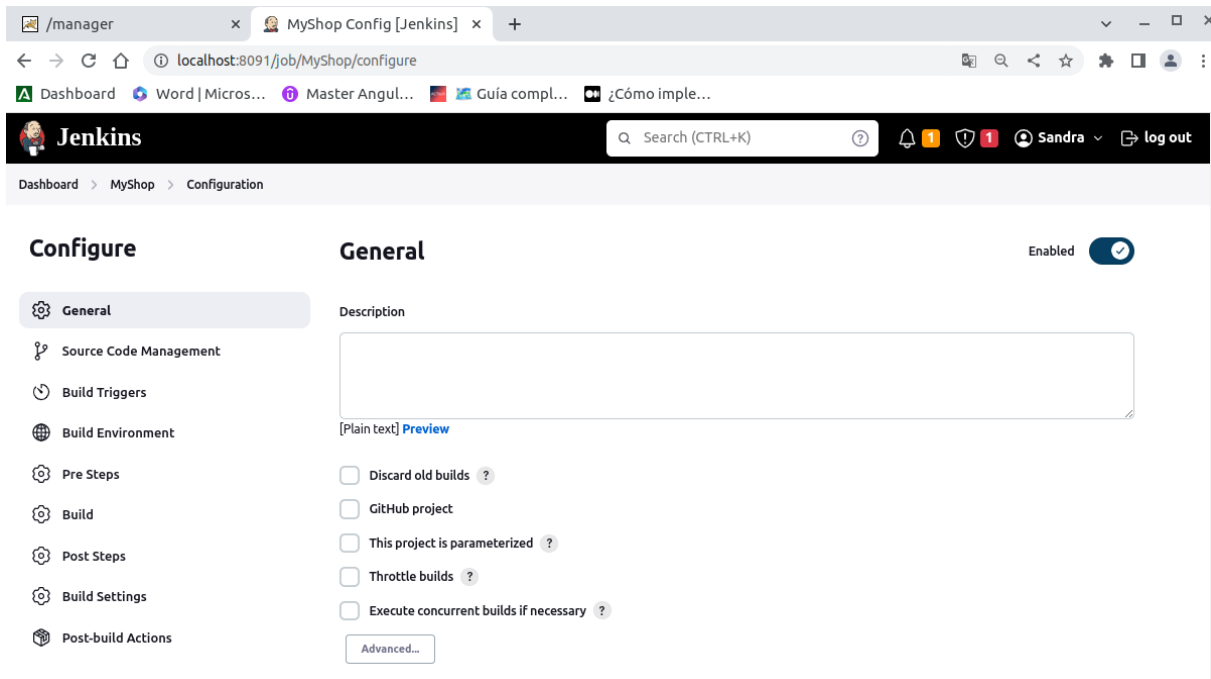
The screenshot shows the 'Maven' configuration page in Jenkins. At the top, there's a section titled 'Maven installations' with a subtitle 'List of Maven installations on this system'. Below this is an 'Add Maven' button. The main configuration area is a dashed box containing a 'Maven' header, a 'Name' field with 'Maven 3.9.0', and a checked 'Install automatically' checkbox. Below the checkbox is a sub-section 'Install from Apache' with a 'Version' dropdown menu set to '3.9.0' and an 'Add Installer' button. Another 'Add Maven' button is at the bottom of the dashed box.

Ya podemos proceder a crear el nuevo proyecto. Seleccionamos en el panel izquierdo el apartado New Item. Le damos un nombre y escogemos la opción Maven project.

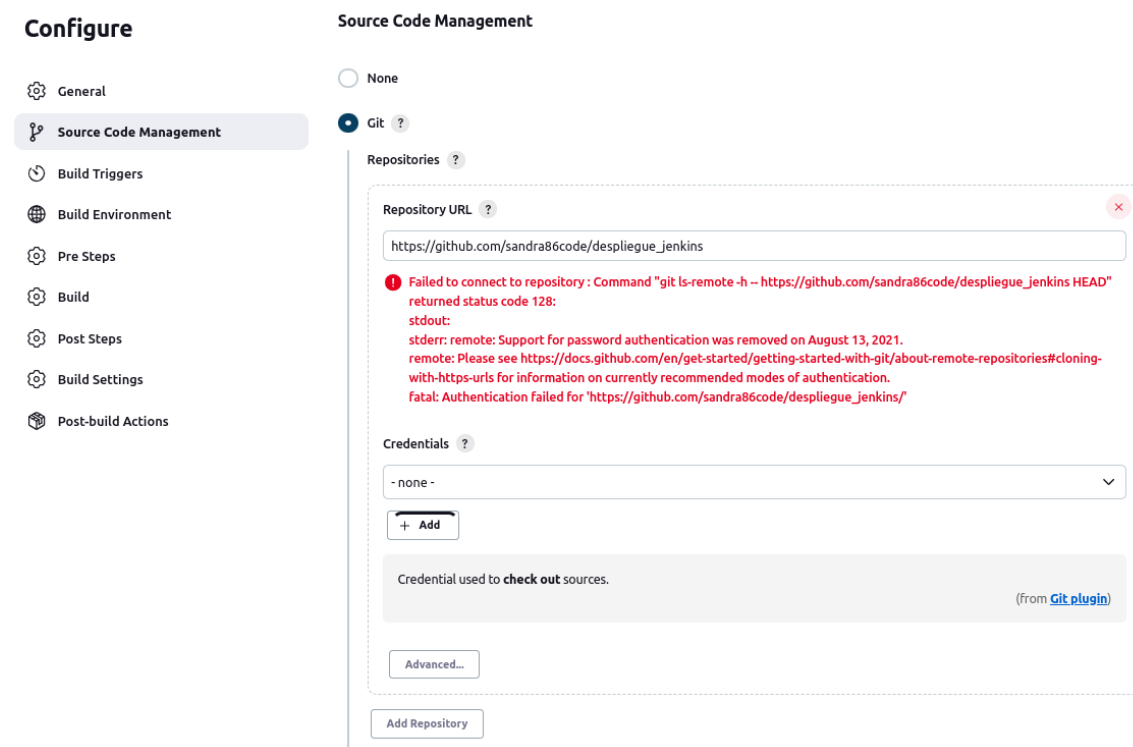
The screenshot shows the 'New Item' dialog in Jenkins. At the top, there's a text input field containing 'MyShop' with a 'Required field' note below it. Below the input field is a list of project types, each with an icon and a description:

- Freestyle project**: This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Maven project**: Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- 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.
- Multi-configuration project**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, 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.
- Multibranch Pipeline**: Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**

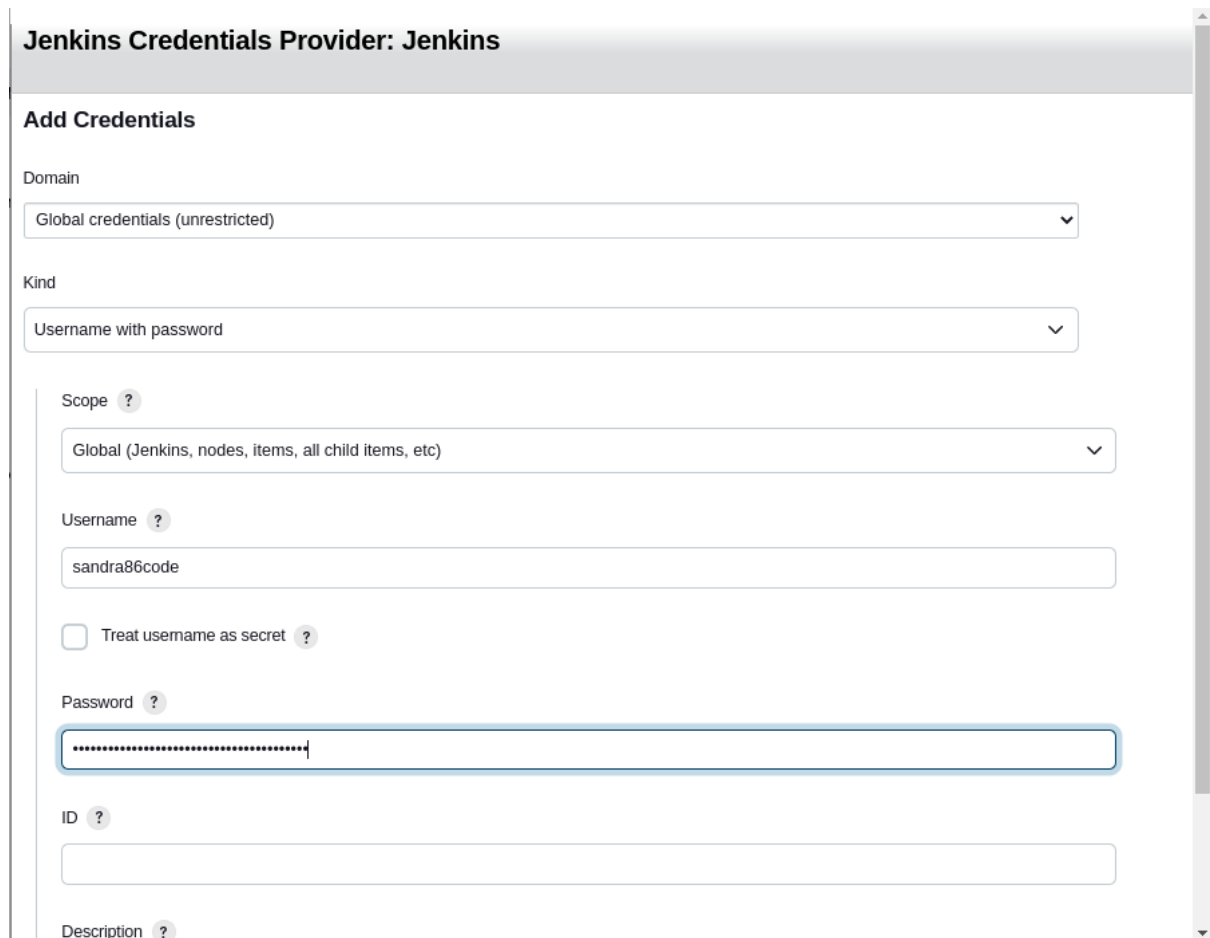
A continuación nos aparece un menú dentro de nuestra nueva tarea:



Lo primero que debemos hacer, es configurar el origen del código fuente, donde debemos indicar a Jenkins la url del repositorio de Git de donde queremos que coja el proyecto a desplegar. Esto lo hacemos en Source Code Management:



Tenemos que añadirle las credentials de GitHub, en este caso elegimos nombre y contraseña, aunque también se puede hacer por SSH.



The screenshot shows the 'Jenkins Credentials Provider: Jenkins' interface. The 'Add Credentials' section is active. The 'Domain' dropdown is set to 'Global credentials (unrestricted)'. The 'Kind' dropdown is set to 'Username with password'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Username' field contains 'sandra86code'. The 'Treat username as secret' checkbox is unchecked. The 'Password' field is filled with dots and is highlighted with a blue border. The 'ID' field is empty. The 'Description' field is empty.

A continuación debemos especificar la rama en la que va a construir. Es importante que sea la misma rama en la que se encuentra alojado el repositorio en GitHub.

El siguiente paso, en el apartado Post-build actions debemos seleccionar la opción “Deploy war/ear to a container”, ya que pretendemos que Jenkins despliegue la aplicación en Tomcat.

Indicamos dónde encontrará el archivo .war:

Post-build Actions

Deploy war/ear to a container

WAR/EAR files ?

target/**/*.war

Context path ?

Containers

Add Container ▾

☐ Deploy on failure

Add post-build action ▾

También indicaremos los datos del contenedor que queremos que use para desplegar la aplicación. En este caso escogemos el Tomcat (versión 10 porque es la que hemos instalado) y en credenciales agregamos las credenciales del rol <manager-script>, que en nuestro caso eran tomcat-tomcat. La url del contenedor de Tomcat la averiguamos ejecutando un docker inspect a dicho contenedor.


```
sandra@sandra-UX410UAK: ~/entornoservidorespring-worksp...
"Links": null,
"Aliases": [
  "cb6750215d4d",
  "tomcat"
],
"NetworkID": "c4041c5684f92713f9da9af91441cbbd37d9f868a2b9cc
cd2c0f735c5d08300f",
"EndpointID": "b6db3cb6b513bfcba36f9d4021c466e3f593d5efa6cc
ef01cec38da501821f7",
"Gateway": "172.24.0.1",
"IPAddress": "172.24.0.4",
"IPPrefixLen": 16,
"IPv6Gateway": "",
"GlobalIPv6Address": "",
"GlobalIPv6PrefixLen": 0,
"MacAddress": "02:42:ac:18:00:04",
"DriverOpts": null
}
}
}
}
]
sandra@sandra-UX410UAK:~/entornoservidorespring-workspace/mitiendaSpringRuizSandr
a2$
```

Containers

Tomcat 9.x Remote

Credentials

- none -

+ Add

Tomcat URL ?

http://172.24.0.4

Advanced...

Add Container

☐ Deploy on failure

Add post-build action

Ya podemos guardar todos los cambios y desplegar nuestra aplicación. La seleccionamos y le damos a construir ahora.

```

[INFO] --- maven-resources-plugin:3.3.0:resources (default-resources) @ miTiendaSpringRuizSandra ---
[INFO] Copying 1 resource
[INFO] Copying 24 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.10.1:compile (default-compile) @ miTiendaSpringRuizSandra ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 26 source files to /var/jenkins_home/workspace/myShop/target/classes
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 2.849 s
[INFO] Finished at: 2023-02-14T15:57:34Z
[INFO] -----
Waiting for Jenkins to finish collecting data
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.10.1:compile (default-compile) on
project miTiendaSpringRuizSandra: Fatal error compiling: error: invalid target release: 17 -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoExecutionException
[JENKINS] Archiving /var/jenkins_home/workspace/myShop/pom.xml to
com.jacaranda/miTiendaSpringRuizSandra/1/miTiendaSpringRuizSandra-1.pom
[DeployPublisher][INFO] Build failed, project not deployed
Finished: FAILURE

```

Como vemos, no se nos ha construido el proyecto, por un problema con los Plugins, muy probablemente, que no se ha podido solventar.

Por ello, lo hemos hecho con un proyecto ya hecho subido en GitHub y sí ha funcionado.

La única cosa que debemos cambiar del proceso que hemos seguido hasta aquí es que, al ser un repositorio público, no necesitamos introducir las credenciales de GitHub.

Si accedemos a la url, podemos ver el proyecto desplegado sobre nuestro contenedor Tomcat.

Soccer

Autor: Wikipedia HD @ <http://webpédia.es>

Teams

GET Teams	GET Team
	POST Team

Stadium

GET Stadium	POST Stadium
-------------	--------------

Players

GET Players	POST Players
	GET Player

BIBLIOGRAFÍA

García, Roberto (s.f.). *Tema 5. Administración de servidores de aplicaciones (Tomcat)*. Disponible en: <https://sites.google.com/site/desplieguedeaplicacionesweb/tema-5-administracion-de-servidores-de-aplicaciones-tomcat>. Consultado: 25/01/2023

Jenkins (s.f.). *Jenkins*. Disponible en: <https://www.jenkins.io/>. Consultado: 14/02/2023

Microsoft (s.f.). *Tutorial: Creación de aplicaciones de varios contenedores con MySQL y Docker Compose*. Disponible en: <https://learn.microsoft.com/es-es/visualstudio/docker/tutorials/tutorial-multi-container-app-mysql>. Consultado: 25/01/2023

programador clic (s.f.). *Error al iniciar el proyecto de arranque Spring: java.lang.IllegalArgumentException: LoggerFactory no es un Logback*. Disponible en: <https://programmerclick.com/article/4892741451/>. Consultado: 14/02/2023

programador clic (s.f.). *Obtenga la ruta de referencia css y js en SpringMVC*. Disponible en: <https://programmerclick.com/article/2068802451/>. Consultado: 14/02/2023

StackOverflow (2015). *Jenkins CLI connection refused*. Disponible en: <https://stackoverflow.com/questions/30502076/jenkins-cli-connection-re>. Consultado: 14/02/2023