**CONFIGURACIÓN DE AMBIENTE**

**(LINUX)**

**Instalar Java 19**

Verificamos que se tenga instalado java con el comando: java -version

apt install openjdk-19-jre-headless

Si no está instalado, ejecutamos el comando:

wget https://download.oracle.com/java/19/latest/jdk-19\_linux-x64\_bin.tar.gz

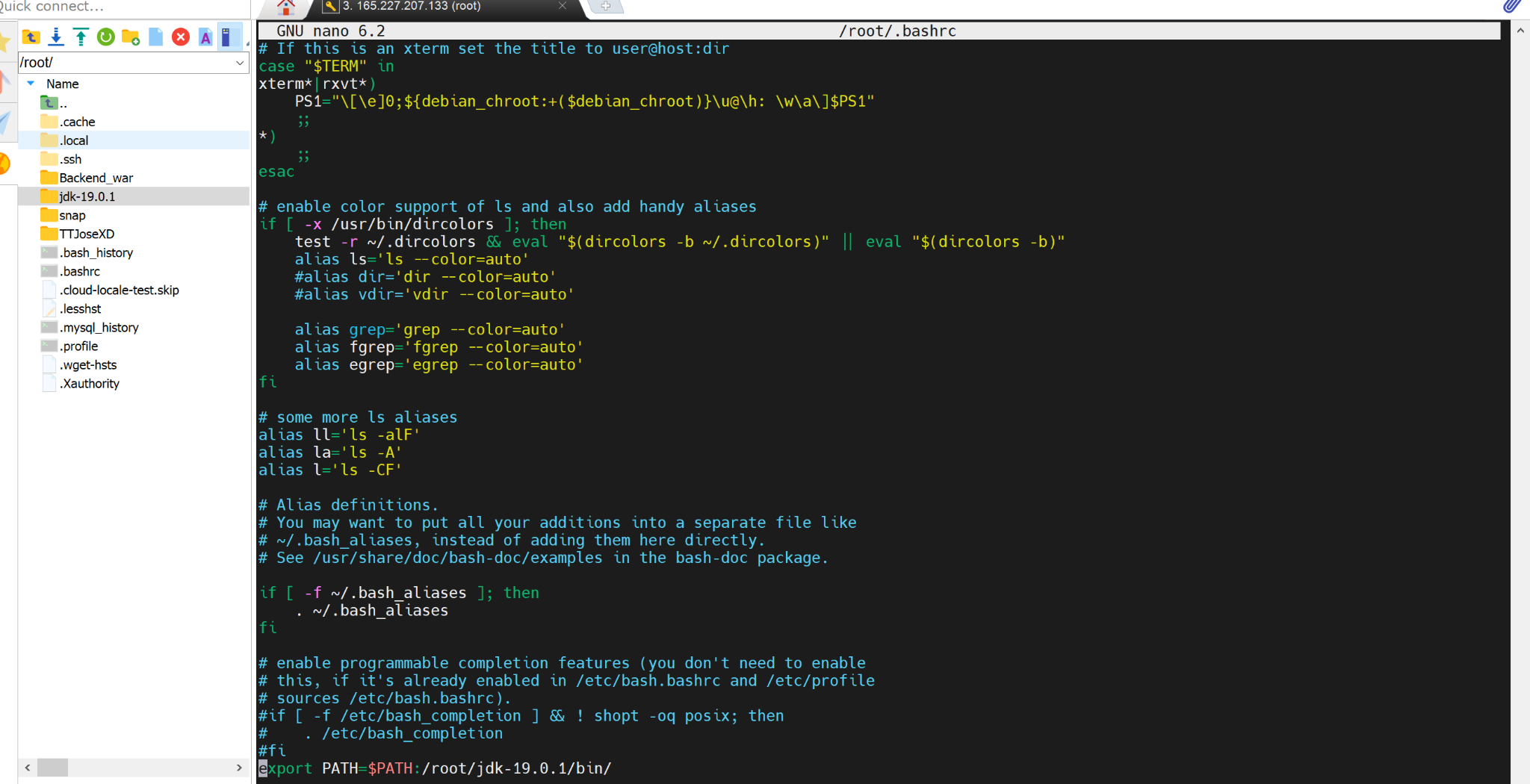
Ejecutamos el comando:

tar xvf jdk-19\_linux-x64\_bin.tar.gz

Dentro de la carpeta jdk-19.0.1, en bin editamos el archivo:

nano /root/.bashrc

y al final del archivo agregamos la linea:



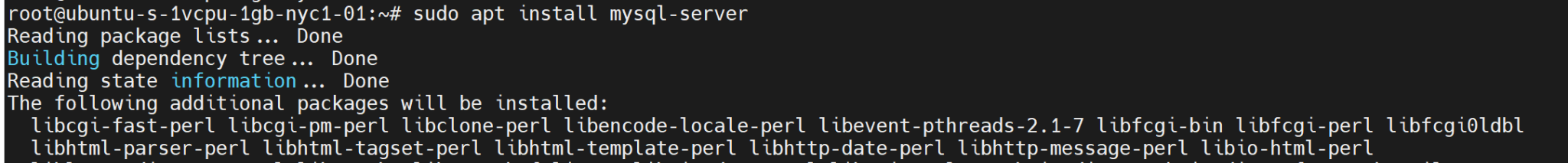
/lib/jvm/java-19-openjdk-amd64/bin/

Otro: export $PATH=rutajava/bin:$PATH

****

**Instalación de MySql**

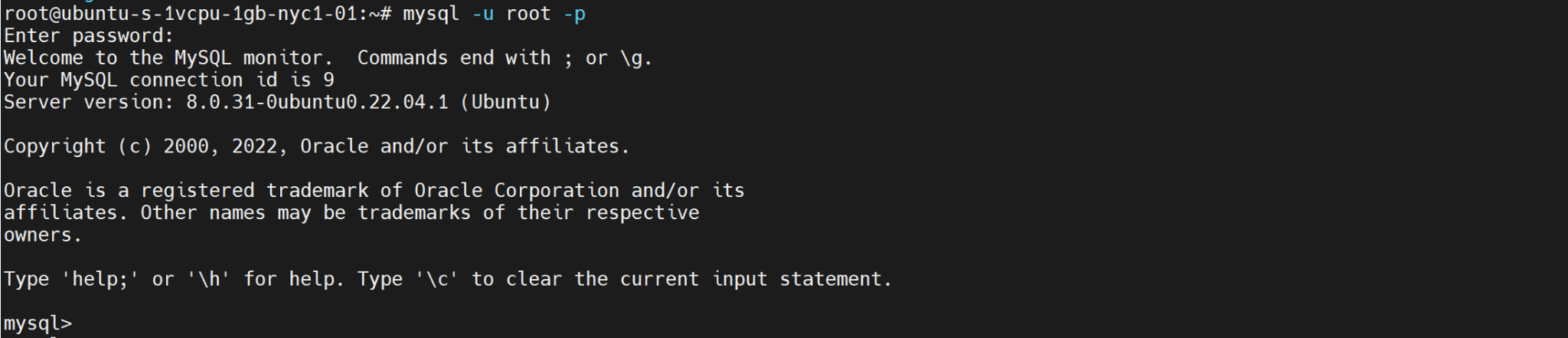
sudo apt install mysql-server



Una vez instalado mysql hay que crear una contraseña para el usuario root

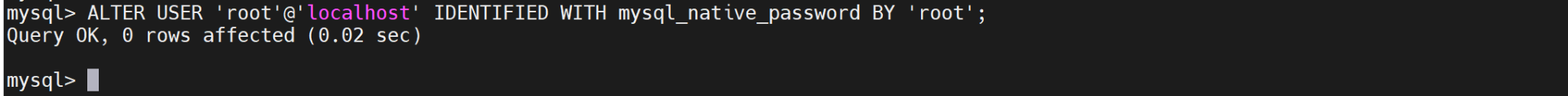
mysql -u root -p

E introducimos la contraseña root.



Si entra sin requerir contraseña, ejecutamos:

ALTER USER ‘root’@’localhost’ IDENTIFIED WITH mysql\_native\_password BY ‘root’;



Crear base de datos: create database seccionb;

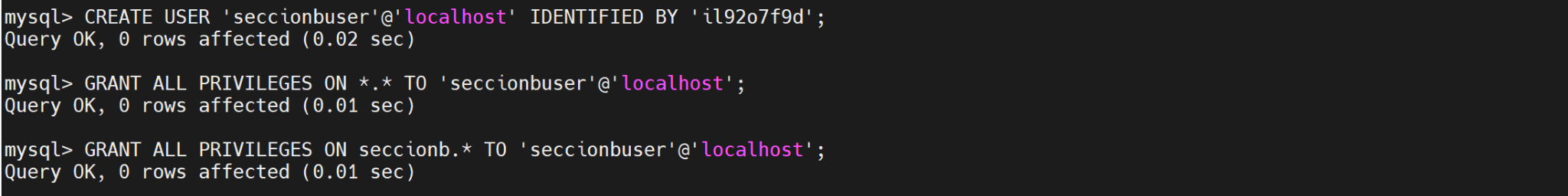


Creamos el usuario y concedemos privilegios sobre la base de datos:

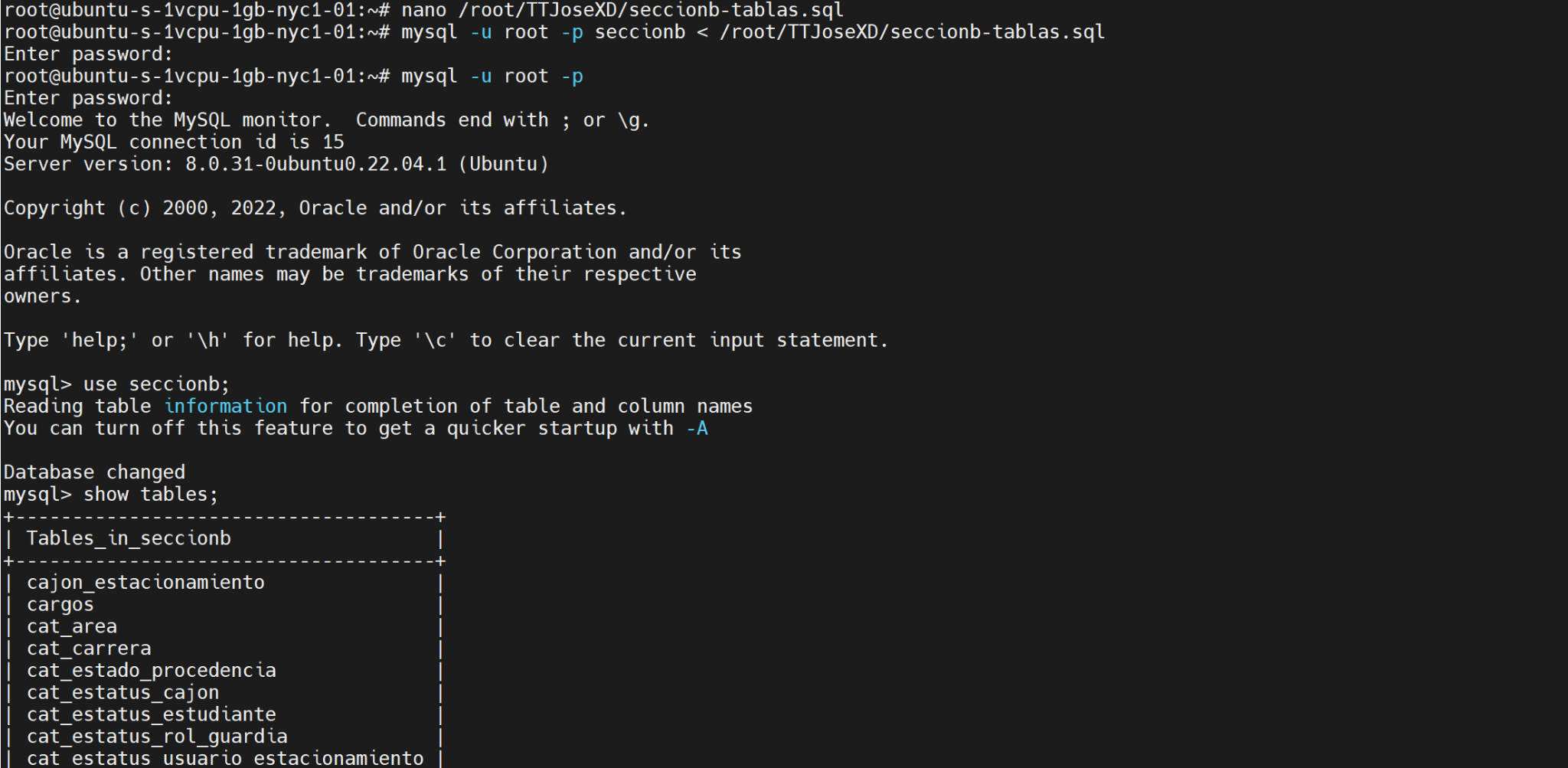
CREATE USER 'seccionbuser'@'localhost' IDENTIFIED BY 'il92o7f9d';

GRANT ALL PRIVILEGES ON \*.\* TO 'seccionbuser'@'localhost';

GRANT ALL PRIVILEGES ON seccionb.\* TO 'seccionbuser'@'localhost';



Copiamos el script de base de datos al servidor y desde la terminal ejecutamos el comando para crear las tablas en la base de datos:

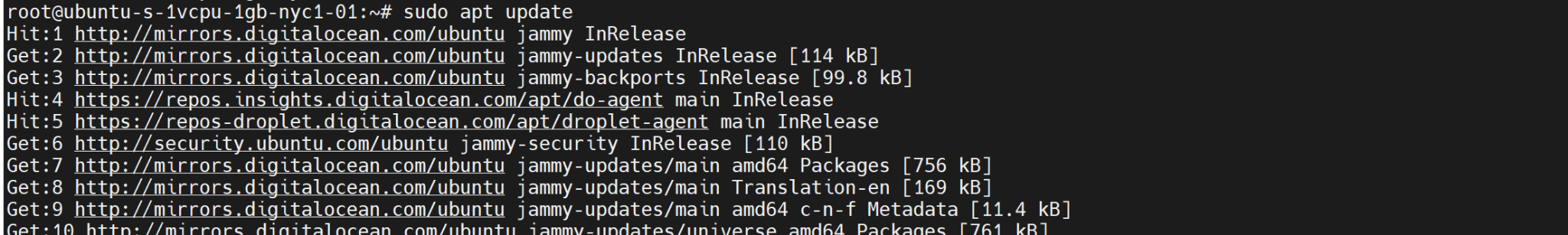


wget <https://download1584.mediafire.com/lk5oo1dtzyyg/mwyjefmhrcgcehp/seccionb-tablas.sql>

mysql -u root -p seccionb < /home/jose/Descargas/seccionb-tablas.sql

**Instalación Wildfly**

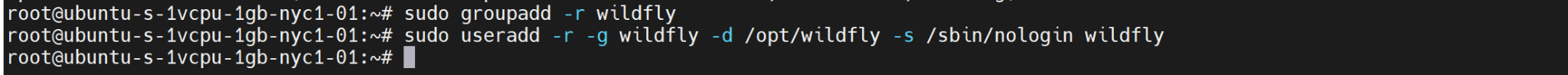
Primero se debe actualizar el servidor: sudo apt update



Dado que este es un servidor de aplicaciones, generalmente **se recomienda usar una cuenta de servicio dedicada**. Ejecuta en la terminal los siguientes comandos para crear una cuenta de servicio de WildFly para administrar el servidor:

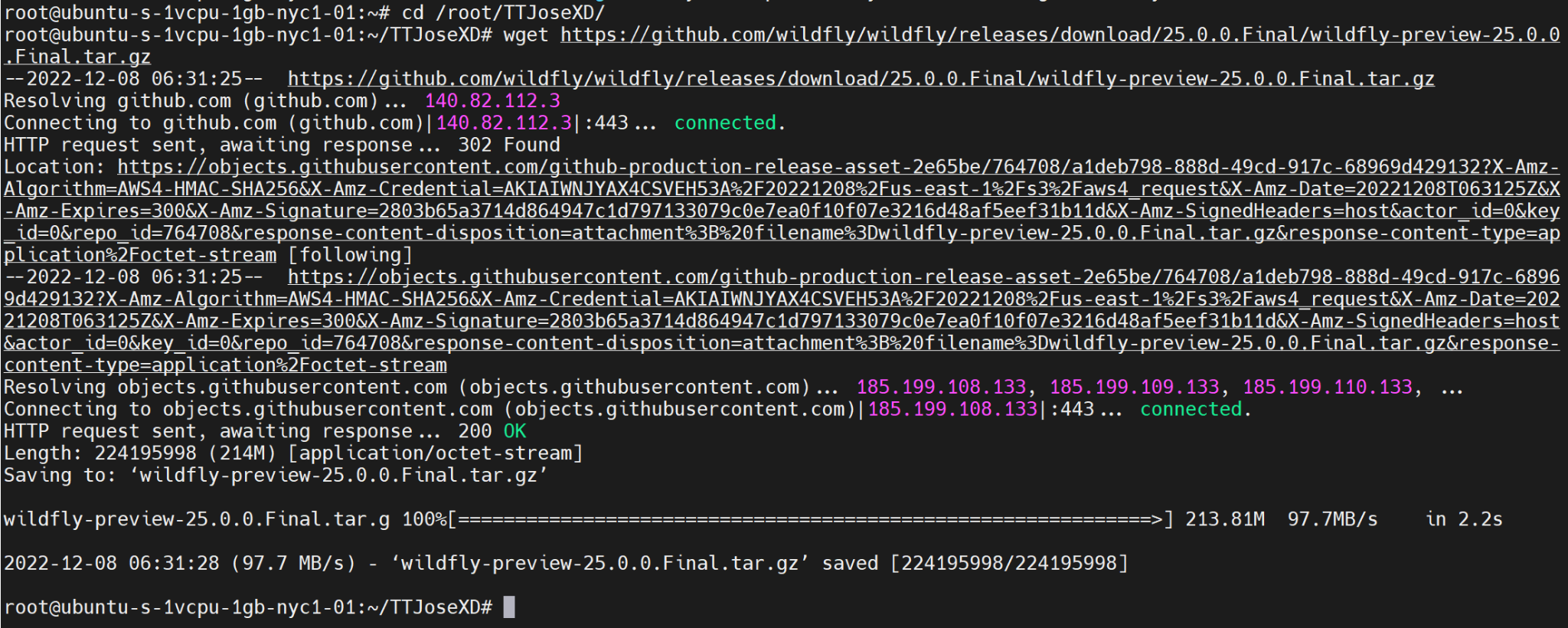
sudo groupadd -r wildfly

sudo useradd -r -g wildfly -d /opt/wildfly -s /sbin/nologin wildfly



Seleccionamos una carpeta donde descargamos wildfly. Dentro ejecutamos el comando:

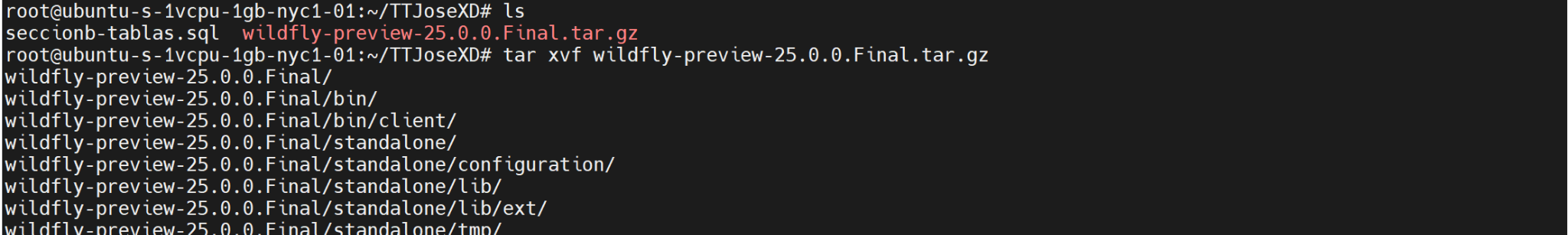
wget <https://github.com/wildfly/wildfly/releases/download/25.0.0.Final/wildfly-preview-25.0.0.Final.tar.gz>



Terminada la descarga, tendremos que ejecutar los siguientes comandos para **crear la carpeta WildFly en el directorio */opt* y para cambiar su propiedad a la cuenta de servicio de WildFly**.

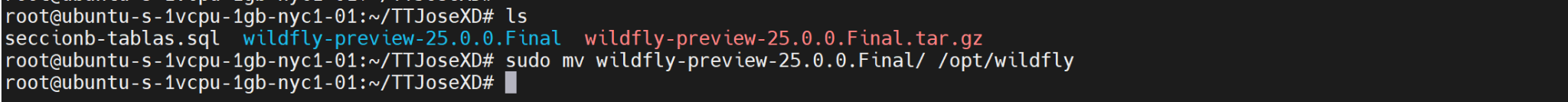
Descomprimimos el archivo descargado:

tar xvf wildfly-preview-25.0.0.Final.tar.gz

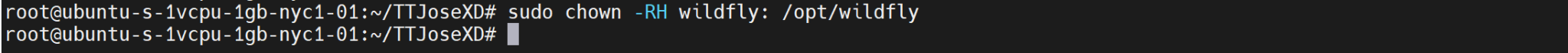


Movemos la carpeta descomprimida a /opt/

sudo mv wildfly-preview-25.0.0.Final/ /opt/wildfly



sudo chown -RH wildfly: /opt/wildfly



A continuación, configuraremos WildFly para que se ejecute como un servicio. Comenzaremos creando un directorio '/ etc / wildfly'.

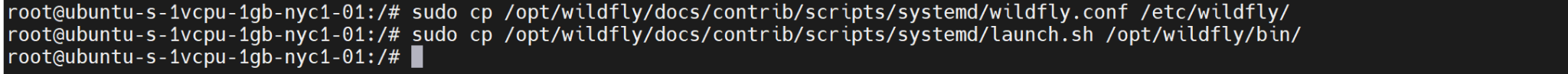
sudo mkdir -p /etc/wildfly



Tras crear el directorio, **copia los archivos de configuración de WildFly y los ejecutables en el directorio** recién creado.

sudo cp /opt/wildfly/docs/contrib/scripts/systemd/wildfly.conf /etc/wildfly/

sudo cp /opt/wildfly/docs/contrib/scripts/systemd/launch.sh /opt/wildfly/bin/



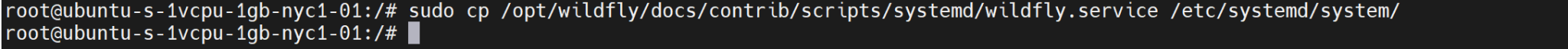
También tenemos que **hacer que los scripts en el directorio */etc/wildfly/bin* sean ejecutables**.

sudo sh -c 'chmod +x /opt/wildfly/bin/\*.sh'



En este punto vamos a **copiar el archivo systemd al directorio */etc/systemd/system/*** ejecutando:

sudo cp /opt/wildfly/docs/contrib/scripts/systemd/wildfly.service /etc/systemd/system/



Ahora podemos **utilizar los siguientes comandos para detener, iniciar y habilitar los servicios de WildFly** para que se inicien automáticamente en el momento del arranque:

**Detener el servicio**:

sudo systemctl stop wildfly.service

**Iniciar el servicio:**

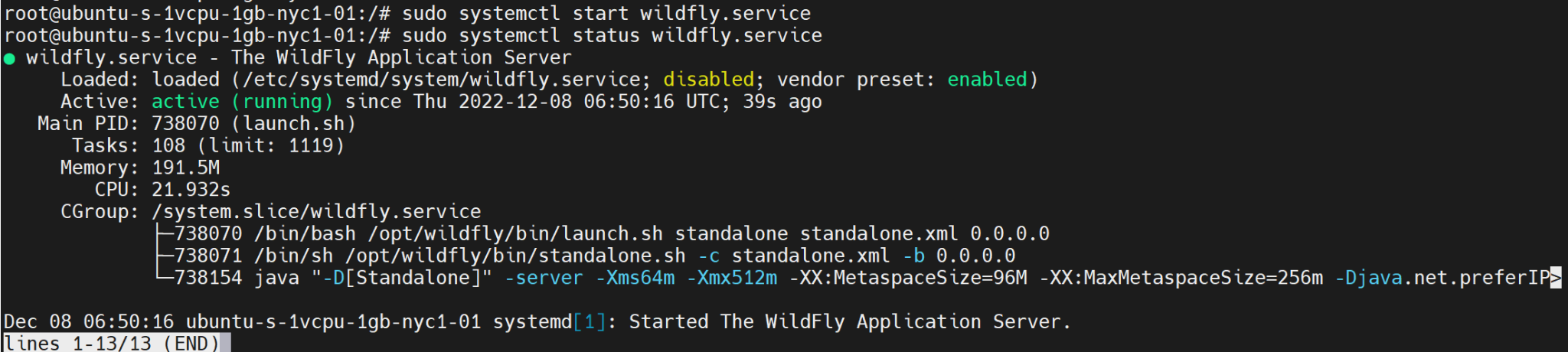
sudo systemctl start wildfly.service

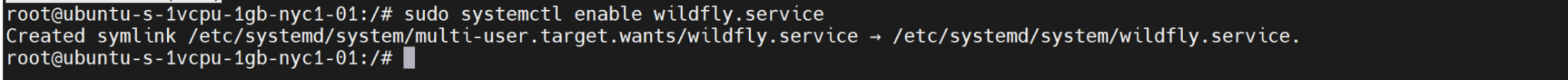
**Habilitar el servicio:**

sudo systemctl enable wildfly.service

Para **verificar el inicio**, ejecuta el siguiente comando:

sudo systemctl status wildfly.service





## **Cuenta de usuario para conectar y administrar la consola web del servidor**

sudo /opt/wildfly/bin/add-user.sh



**Cambiar puerto por default al 8181 en wildfly**

Debemos editar el archivo standalone.xml:

nano /etc/wildfly/standalone/configuration/standalone.xml

Buscamos el tag <socket-binding-group>

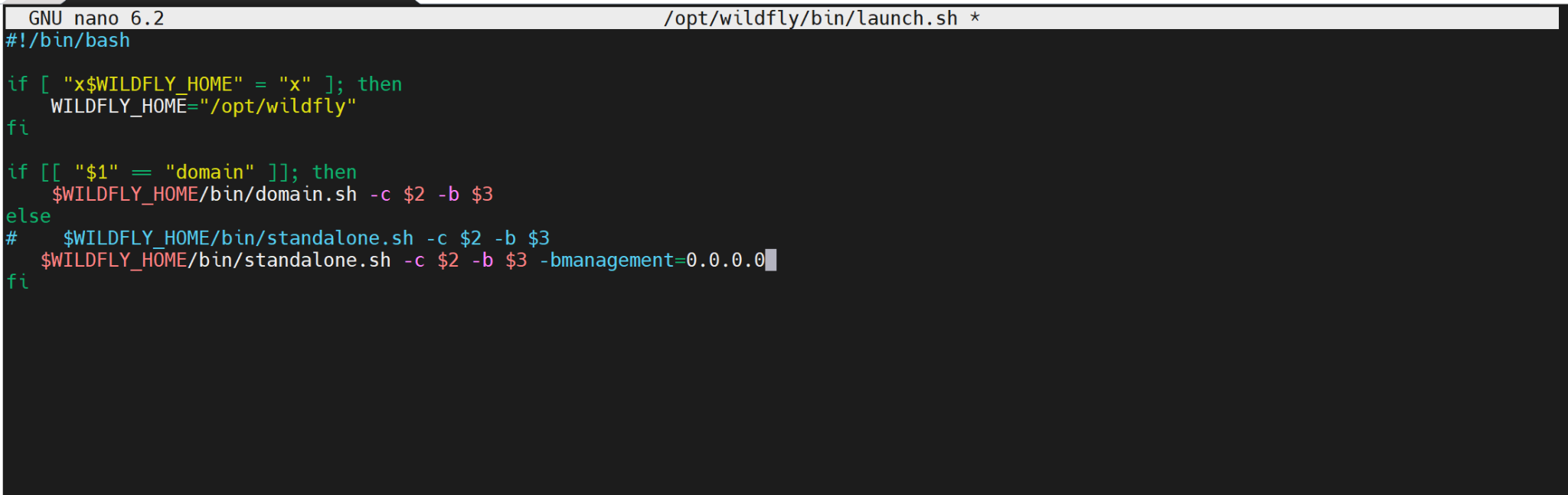
y dentro modificamos la linea <socket-binding name="http" port="${jboss.http.port:8181}"/>

cambiamos 8080 por 8181.

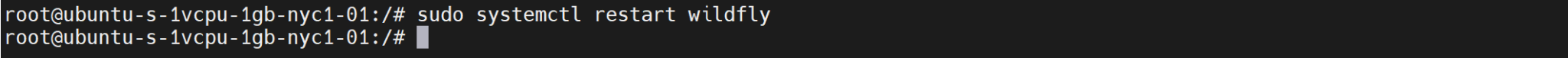
**Configuración para aceptar conexiones externas:**

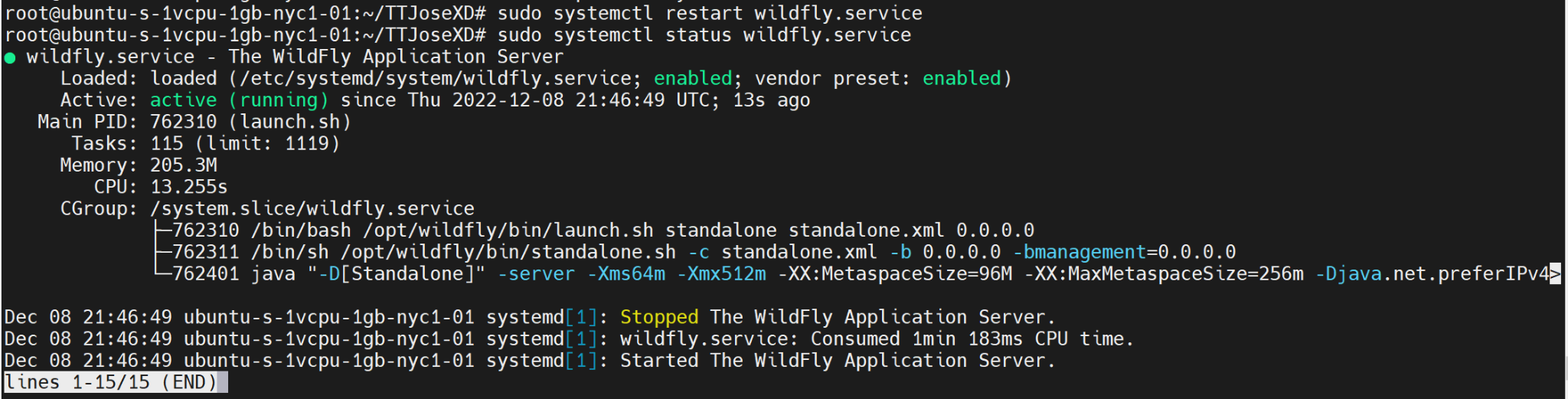
Editamos el archivo lauch.sh y modificamos las lineas como se muestran a continuación:

nano /opt/wildfly/bin/launch.sh

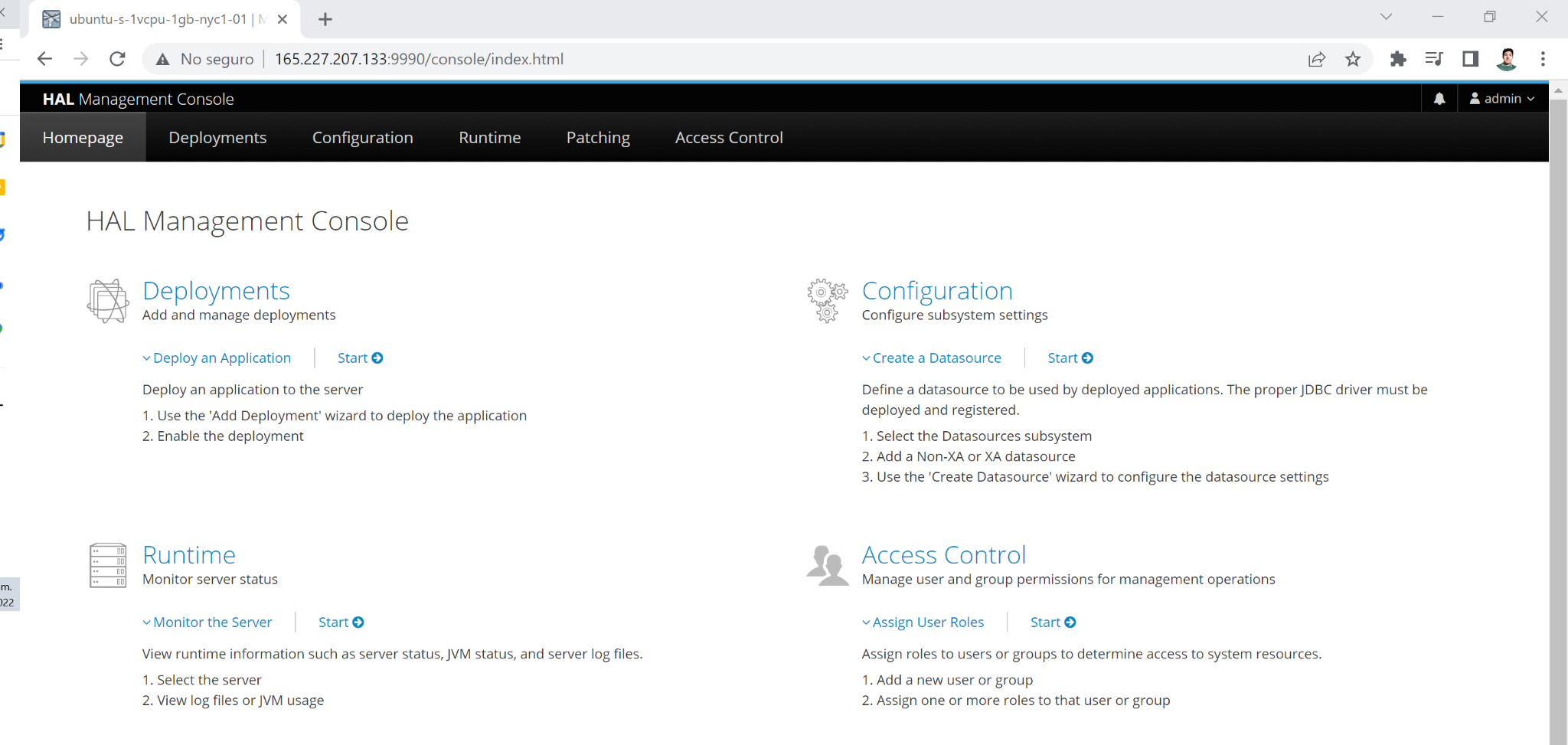


Guardamos y reiniciamos el servicio:





Para este punto, si deseamos ingresar a la consola de administración de wildfly desde nuestro equipo local lo podemos hacer ingresando desde la ip del servidor:

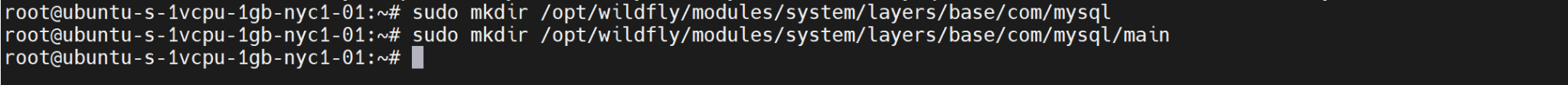


**Configurar conector de base de datos**

Lo primero es agregar al SA el módulo para trabajar con MySql, esto se hace usando el archivo jar del driver de la base de datos mysql-connector-java-8.0.30.jar, para eso se realizará lo siguiente:

Desde el servidor, crear el directorio:

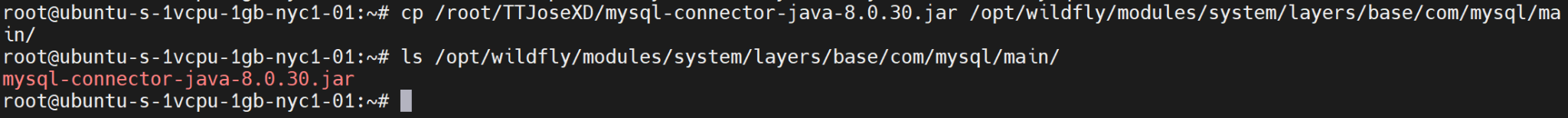
sudo mkdir /opt/wildfly/modules/system/layers/base/com/mysql/main



Copiar el mysql-connector-java-8.0.30.jar a la carpeta main:

wget <https://download1528.mediafire.com/mqhkhspdy8kg/ho0ow292j9lxaod/mysql-connector-java-8.0.30.jar>

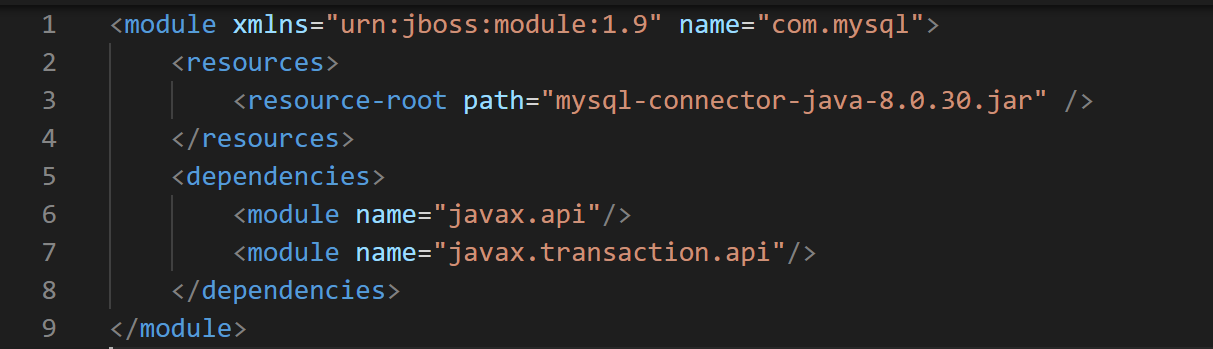
sudo cp /root/TTJoseXD/mysl-connector-java-8.0.30.jar /opt/wildfly/modules/system/layers/base/com/mysql/main



ahora se debe crear dentro de la misma carpeta el archivo module.xml con el siguiente contenido:

nano module.xml (intentar gedit)

|  |
| --- |
| <module xmlns="urn:jboss:module:1.9" name="com.mysql">  <resources>  <resource-root path="mysql-connector-java-8.0.30.jar" />  </resources>  <dependencies>  <module name="javax.api"/>  <module name="javax.transaction.api"/>  </dependencies>  </module> |



**Configuración datasource en standalone.xml**

Es necesario realizar modificaciones al archivo standalone.xml que se encuentra en: /opt/wildfly/standalone/configuration/standalone.xml.

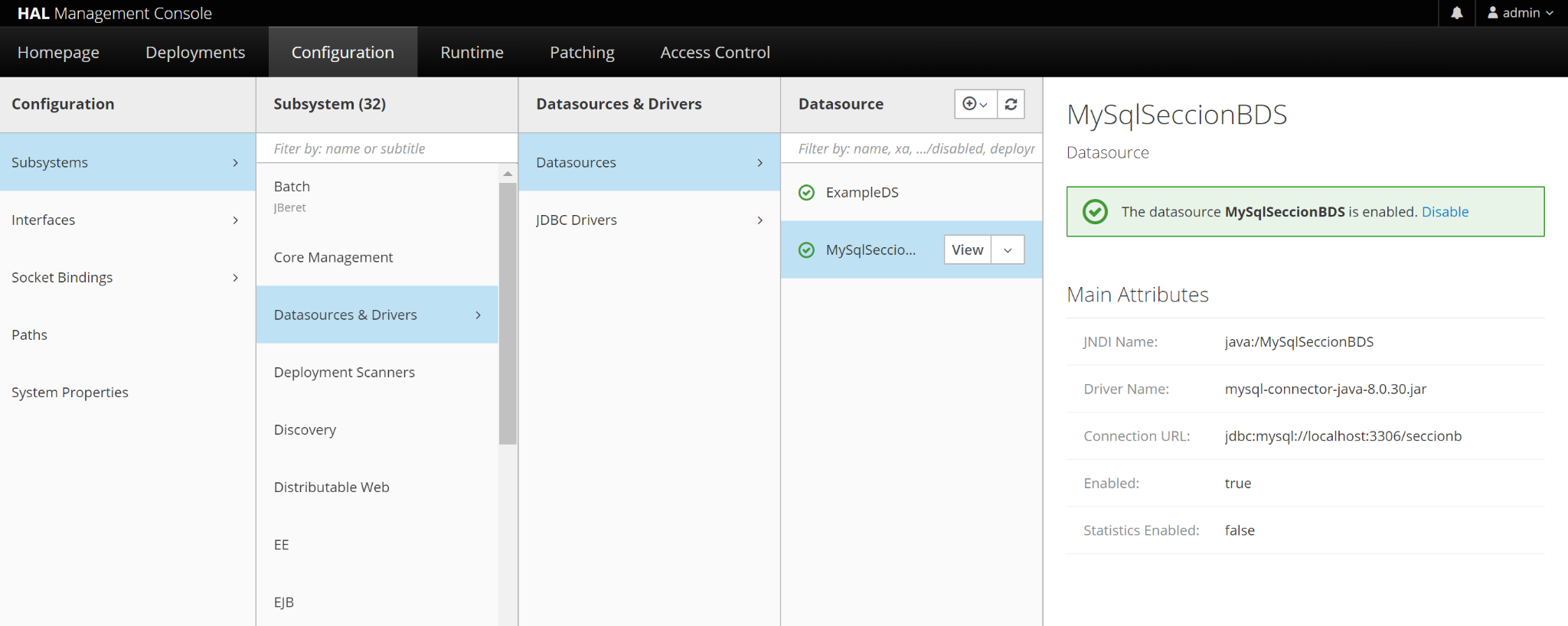
Agregar al tag de <drivers> las líneas:

|  |
| --- |
| <driver name="mysql" module="com.mysql">  <driver-class>com.mysql.cj.jdbc.Driver</driver-class>    <xa-datasource-class>com.mysql.cj.jdbc.MysqlXADataSource</xa-datasource-class>  </driver> |

Buscar el tag <datasources> y agregar el siguiente nodo:

|  |
| --- |
| <datasource jta="true" jndi-name="java:jboss/MySqlSeccionBDS" pool-name="MySqlSeccionBDS">  <connection-url>jdbc:mysql://localhost:3306/seccionb</connection-url>  <driver-class>com.mysql.cj.jdbc.Driver</driver-class>  <driver>mysql</driver>  <security>  <user-name>seccionbuser</user-name>  <password>il92o7f9d</password>  </security>  <validation>  <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"/>  <validate-on-match>true</validate-on-match>  <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>  </validation>  </datasource> |

Con lo que si accedemos a la consola de wildfly debemos encontrar el datasource de la siguiente manera:



Buscar el tag <default-security-domain value="other"/>,inmediatamente después se encuentra el tag <application-security-domains>, se debe agregar dentro:

|  |
| --- |
| <application-security-domain name="SeccionbApplicationDomain" security-domain="SeccionbDomain"/> |

La definición de seguridad se hace en el subsistema Elytron <subsystem xmlns="urn:wildfly:elytron:14.0" final-providers="combined-providers" disallowed-providers="OracleUcrypto"> donde se encuentra el tag <security-domains>,dentro del tag agregar:

|  |
| --- |
| <security-domain name="SeccionbDomain" default-realm="SeccionbRealm" permission-mapper="default-permission-mapper">  <realm name="SeccionbRealm" role-decoder="from-roles-attribute"/>  </security-domain> |

Dentro del mismo subsistema, dentro del tag <security-realms>, agregar:

|  |
| --- |
| <jdbc-realm name="SeccionbRealm">  <principal-query sql=" SELECT u.contrasenia FROM usuarios u WHERE u.usuario = ?" data-source="MySqlSeccionBDS">  <simple-digest-mapper algorithm="simple-digest-sha-256" password-index="1"/>  </principal-query>  <principal-query sql=" SELECT r.nombre, 'Roles'  FROM usuarios u INNER JOIN rel\_usuarios\_roles ur ON ur.usuario\_id=u.usuario\_id  INNER JOIN cat\_roles r ON ur.rol\_id=r.rol\_id WHERE u.usuario=? AND u.estatus=1" data-source="MySqlSeccionBDS">  <attribute-mapping>  <attribute to="roles" index="1"/>  </attribute-mapping>  </principal-query>  </jdbc-realm> |

En el mismo subsistema, dentro del tag <mappers>, agregar:

Esto se puede hacer después del tag <simple-role-decoder …/>

|  |
| --- |
| <simple-role-decoder name="from-roles-attribute" attribute="roles"/> |

Un poco más abajo de <mappers>, se encuentra el tag <http>

dentro de ese nodo, agregar:

|  |
| --- |
| <http-authentication-factory name="seccionb-db-http-authentication" security-domain="SeccionbDomain" http-server-mechanism-factory="global">  <mechanism-configuration>  <mechanism mechanism-name="FORM">  <mechanism-realm realm-name="SeccionbRealm"/>  </mechanism>  </mechanism-configuration>  </http-authentication-factory> |

Ahora que ya se definió el método de autenticacion HTTP, en el subsistema <subsystem xmlns="urn:jboss:domain:undertow:12.0" …>

se encuentra el nodo <handlers>

inmediatamente después del nodo de cierre </handlers>, agregar dentro del tag <application-security-domains>:

|  |
| --- |
| <application-security-domain name="SeccionbApplicationDomain" http-authentication-factory="seccionb-db-http-authentication"/> |

**Configuración envío de correos**

Debemos editar el archivo standalone.xml

Eliminamos las líneas:

|  |
| --- |
| <mail-session name="default" jndi-name="java:jboss/mail/Default">  <smtp-server outbound-socket-binding-ref="mail-smtp"/>  </mail-session> |

y las reemplazamos por las siguientes:

Contraseña: **bapf tntd ktpz ahoc**

|  |
| --- |
| <mail-session name="gmailSalida" jndi-name="java:jboss/mail/gmailSalida">  <smtp-server outbound-socket-binding-ref="gmail-salida-sock" ssl="true" username="smtppruebas98@gmail.com" password="bapftntdktpzahoc"/>  </mail-session> |

Borramos la líneas:

|  |
| --- |
| <outbound-socket-binding name="mail-smtp">  <remote-destination host="${jboss.mail.server.host:localhost}" port="${jboss.mail.server.port:25}"/>  </outbound-socket-binding> |

Y las reemplazamos por:

|  |
| --- |
| <outbound-socket-binding name="gmail-salida-sock">  <remote-destination host="smtp.gmail.com" port="587"/>  </outbound-socket-binding> |

**Desplegar proyecto war**

Para desplegar la aplicación web se debe copiar el archivo .war a la carpeta deployments

WAR: <https://download1487.mediafire.com/dpv7wcayxrug/jnu5cfzyspfuxti/webapp-seccionb.war>

