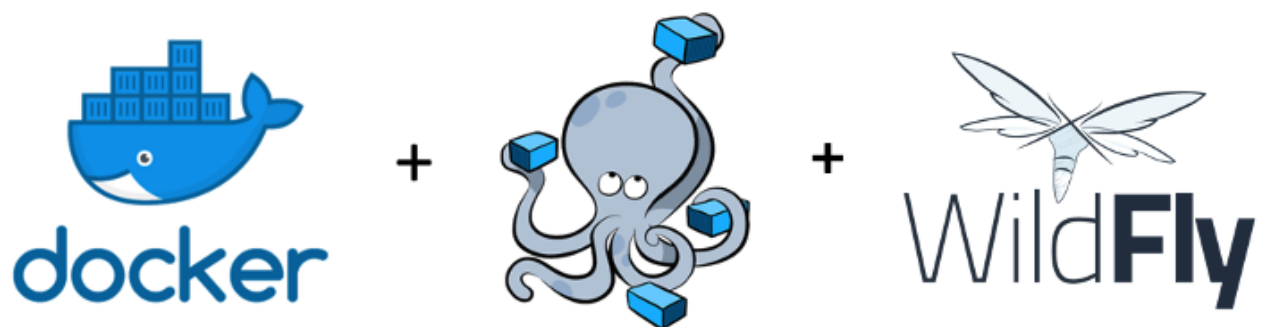


Clusterizando una app en Wildfly en Linux



Jesús Joel Meneses Meneses
2º DAW A
DPL---Despliegue de Aplicaciones Web

Índice

1. Instalación de Docker Compose

. 1.1 Requisitos para instalar Docker Compose

- ## 1.2 Instalar Docker Compose en linux ##

2. Desplegando un Cluster de JBOSS con Docker

- ## 2.1 Construcción del proyecto ##

1. Instalación de Docker Compose

1.1 Requisitos para instalar Docker Compose

Tener instalado Docker en nuestro sistema operativo

1.2 Instalar Docker Compose en Linux

Para descargar la versión estable actual de Docker Compose, ejecutamos el siguiente comando

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

```
joel@joel-VirtualBox:~$ sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
[sudo] contraseña para joel:
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100  633    100  633    0     0   1984      0 --:--:-- --:--:-- --:--:--  1984
100 12.1M   100 12.1M    0     0 7342k      0 0:00:01 0:00:01 --:--:-- 12.1M
```

Posteriormente aplicamos permisos ejecutables al archivo binario

```
sudo chmod +x /usr/local/bin/docker-compose
```

```
joel@joel-VirtualBox:~$ sudo chmod +x /usr/local/bin/docker-compose
```

Seguidamente crearemos un enlace simbólico, con el siguiente comando

```
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

```
joel@joel-VirtualBox:~$ sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

Para saber si tenemos Docker Compose instalado, ejecutaremos el siguiente comando

```
docker-compose --version
```

```
joel@joel-VirtualBox:~$ docker-compose --version
docker-compose version 1.29.2, build 5becea4c
```

2. Desplegando un Cluster de JBOSS con Docker

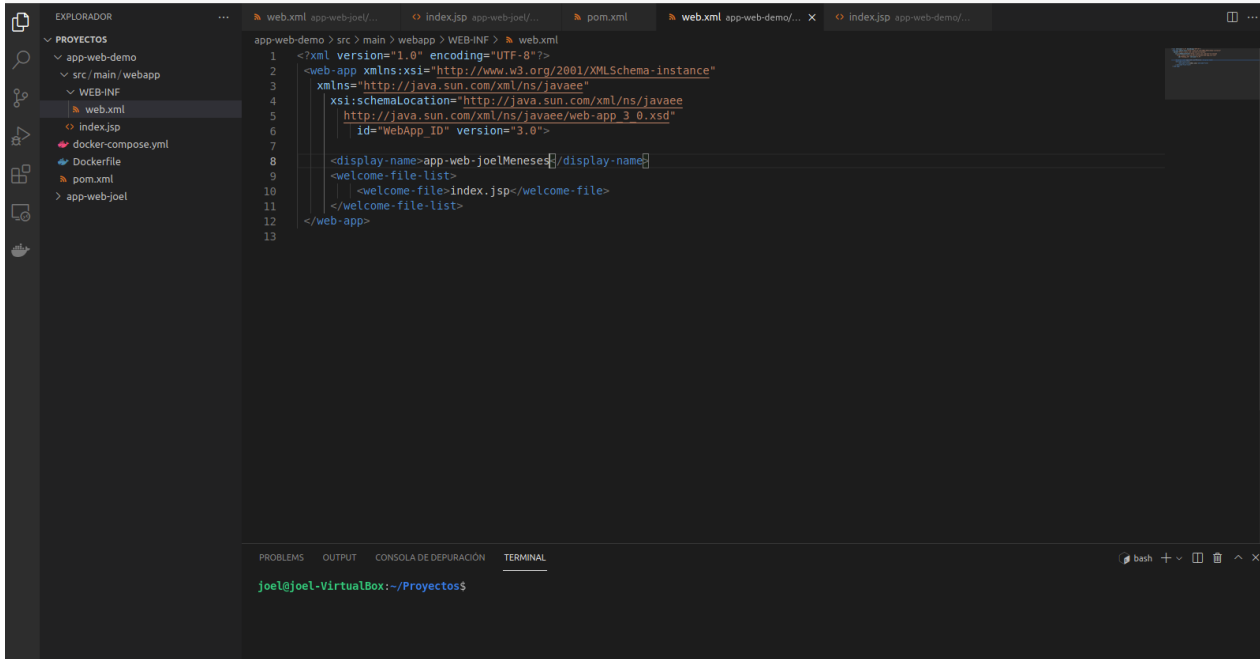
En el siguiente [enlace](#) dispones de un proyecto de una app en Java, donde debes de realizar los siguientes cambios:

Abrimos el proyecto con un editor de código en nuestro caso VSCODE

En el fichero **web.xml** sustituiremos...

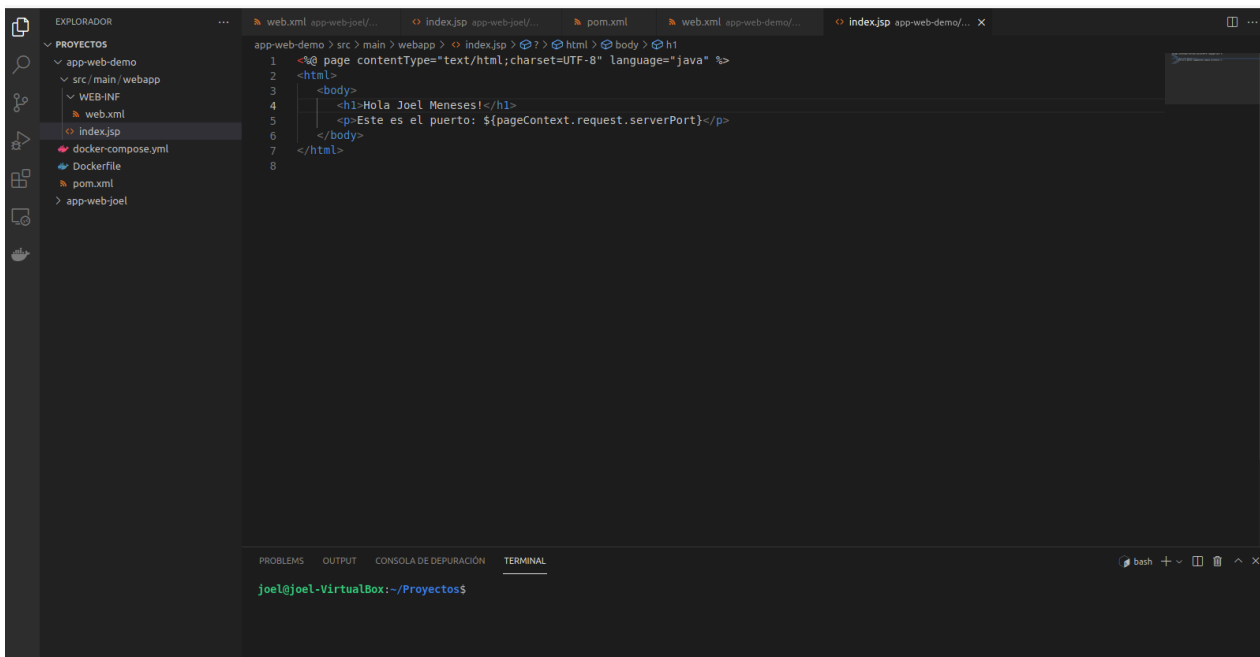
```
app-web-alumno
```

donde alumno seria nuestro alumno



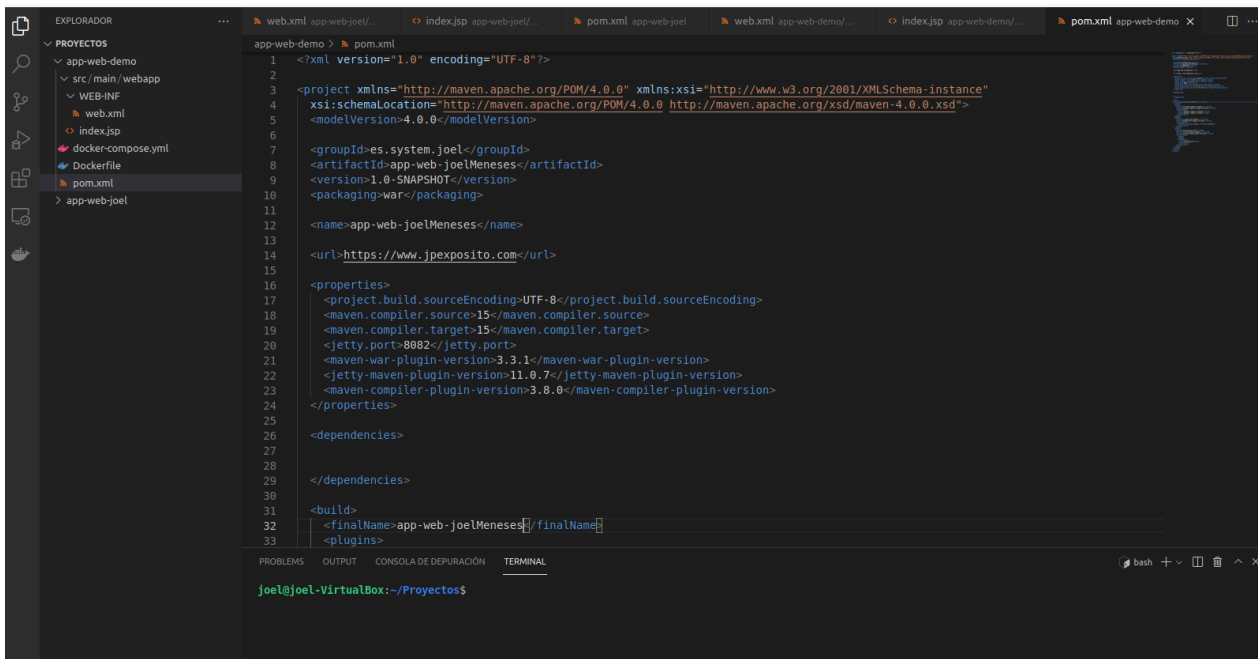
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xmlns="http://java.sun.com/xml/ns/javaee"
4     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5     http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
6     id="WebApp_ID" version="3.0">
7
8     <display-name>app-web-joelMeneses</display-name>
9
10    <welcome-file-list>
11        <welcome-file>index.jsp</welcome-file>
12    </welcome-file-list>
13 </web-app>
```

En el fichero **index.jsp** realizaremos el mismo procedimiento que en el punto anterior

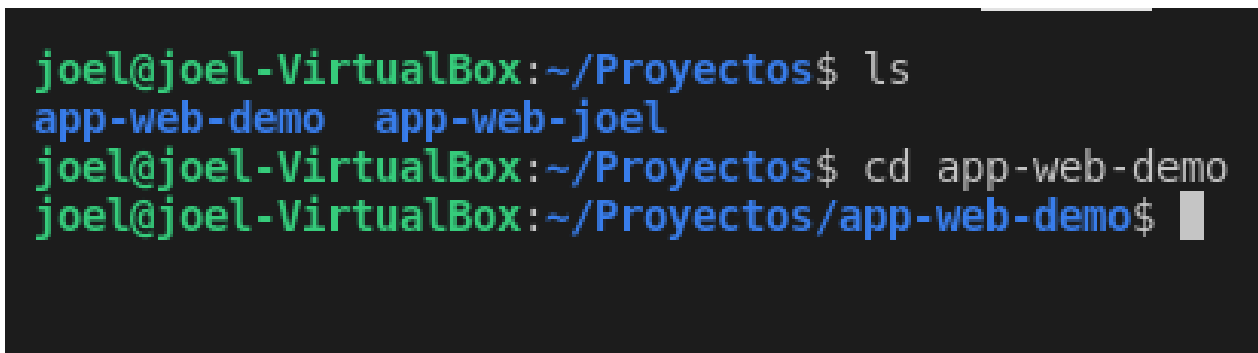


```
1 <%@ page contentType="text/html; charset=UTF-8" language="java" %>
2 <html>
3     <body>
4         <h1>Hola Joel Meneses!</h1>
5         <p>Este es el puerto: ${pageContext.request.serverPort}</p>
6     </body>
7 </html>
```

En el pom sustituiremos el nombre alumno por nuestro nombre



Abriremos la terminal de VSCODE, y añadiremos los comandos que se muestran en la imagen para acceder al proyecto

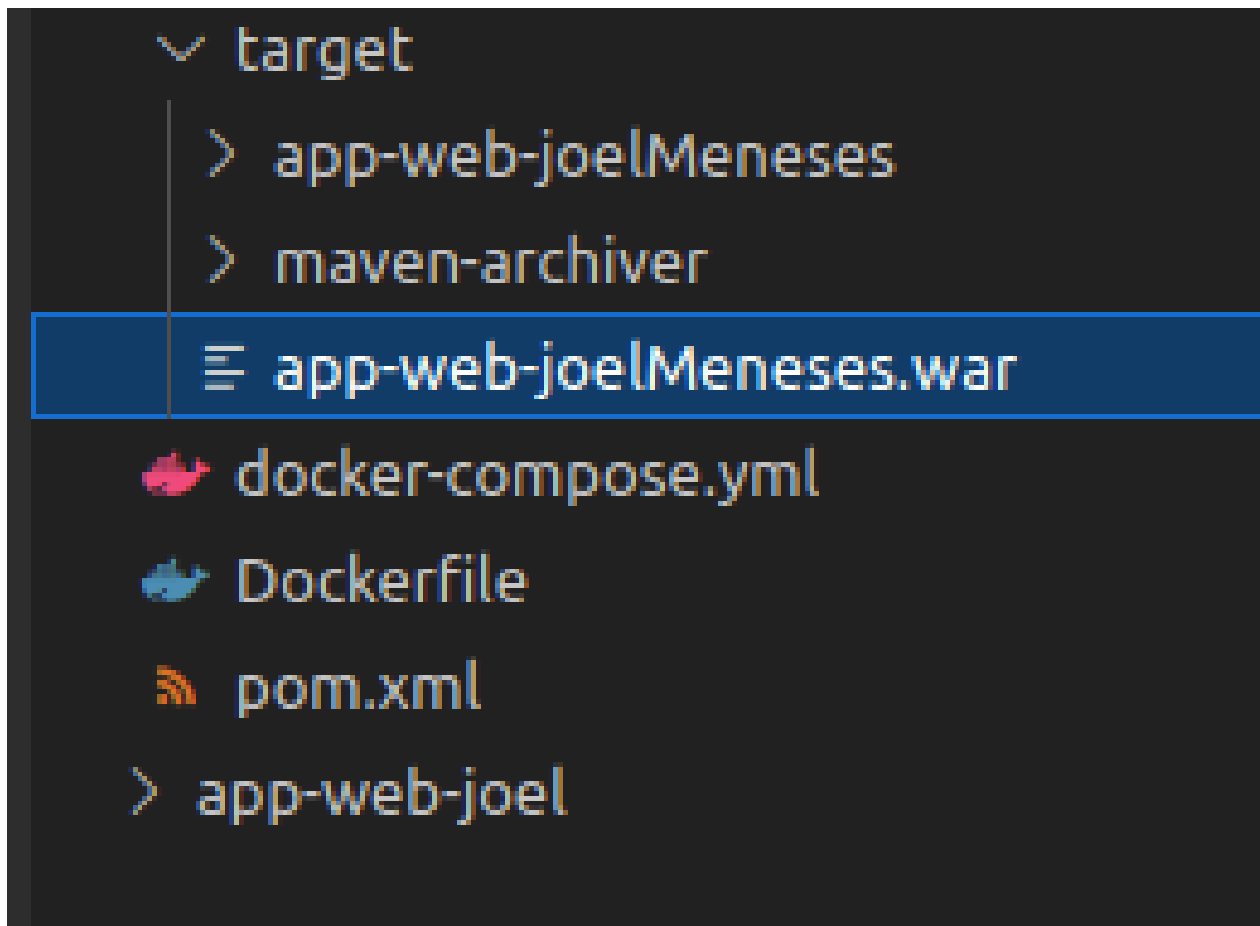


Seguidamente lanzamos el siguiente comando

```
mvn clean install
```

```
joel@joel-VirtualBox:~/Proyectos/app-web-demo$ mvn clean install
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.google.inject.internal.cglib.core.$ReflectUtils$1 (file:/usr/share/maven/lib/guice.jar) to method java.lang.ClassLoader.defineClass(java.lang.String,byte[],int,int,java.security.ProtectionDomain)
WARNING: Please consider reporting this to the maintainers of com.google.inject.internal.cglib.core.$ReflectUtils$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
[INFO] Scanning for projects...
[INFO]
[INFO] -----< es.system.joel:app-web-joeMeneses >-----
[INFO] Building app-web-joeMeneses 1.0-SNAPSHOT
[INFO] -----[ war ]-----
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/maven-clean-plugin-2.5.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/maven-clean-plugin-2.5.jar (3.9 kB at 2.4 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/22/maven-plugins-22.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/22/maven-plugins-22.jar (13 kB at 57 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/21/maven-parent-21.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/21/maven-parent-21.jar (26 kB at 124 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/apache/10/apache-10.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/apache/10/apache-10.jar (15 kB at 89 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/maven-clean-plugin-2.5.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/maven-clean-plugin-2.5.jar (25 kB at 99 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-resources-plugin/2.6/maven-resources-plugin-2.6.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-resources-plugin/2.6/maven-resources-plugin-2.6.jar (8.1 kB at 49 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/23/maven-plugins-23.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/23/maven-plugins-23.jar (9.2 kB at 94 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-resources-plugin/2.6/maven-resources-plugin-2.6.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-resources-plugin/2.6/maven-resources-plugin-2.6.jar (30 kB at 215 kB/s)
```

En la carpeta target veremos el war que nos a creado



Una vez construido el proyecto, podremos ver el resultado ejecutando en modo local el siguiente comando

```
mvn clean jetty:run
```

```
[joel@joel-VirtualBox:~/Proyectos/app-web-demo$ mvn clean jetty:run
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.google.inject.internal.cglib.core.$Refle
(file:/usr/share/maven/lib/guice.jar) to method java.lang.ClassLoader.defineClass(
String,byte[],int,int,java.security.ProtectionDomain)
WARNING: Please consider reporting this to the maintainers of com.google.inject.in
ib.core.$ReflectUtils$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflectiv
perations
WARNING: All illegal access operations will be denied in a future release
[INFO] Scanning for projects...
[INFO]
[INFO] -----< es.system.joel:app-web-joelMeneses >-----
[INFO] Building app-web-joelMeneses 1.0-SNAPSHOT
[INFO] -----[ war ]-----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ app-web-joelMeneses ---
```

Una vez terminada de ejecutar jetty podemos ver el resultado en nuestro navegador escribiendo lo siguiente

```
localhost:8082
```

Nos mostrara un mensaje como el siguiente

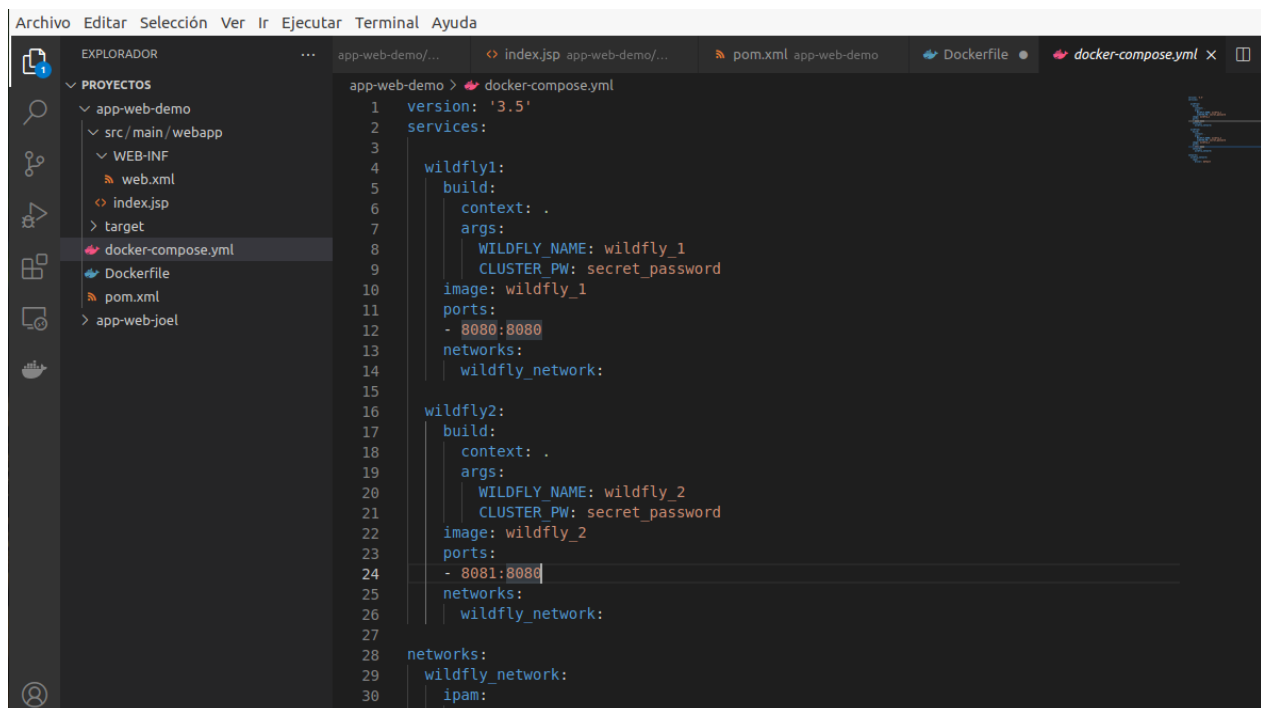


Hola Joel Meneses!

Este es el puerto: 8082

Dentro de la carpeta de nuestra instalación debemos construir el fichero Dockerfile con el contenido siguiente:

```
FROM jboss/wildfly
ARG WAR_FILE=target/*.war
##COPY ${JAR_FILE} app.jar
ADD ${ARG} /opt/jboss/wildfly/standalone/deployments/
ARG WILDFLY_NAME
ARG CLUSTER_PW
ENV WILDFLY_NAME=${WILDFLY_NAME}
ENV CLUSTER_PW=${CLUSTER_PW}
ENTRYPOINT /opt/jboss/wildfly/bin/standalone.sh -b=0.0.0.0 -bmanagement=0.0.0.0 -Djboss.server.default.config=standalone-
full-ha.xml -Djboss.node.name=${WILDFLY_NAME} -Djava.net.preferIPv4Stack=true -Djgroups.bind_addr=$(hostname -i) -
Djboss.messaging.cluster.password=${CLUSTER_PW}
```



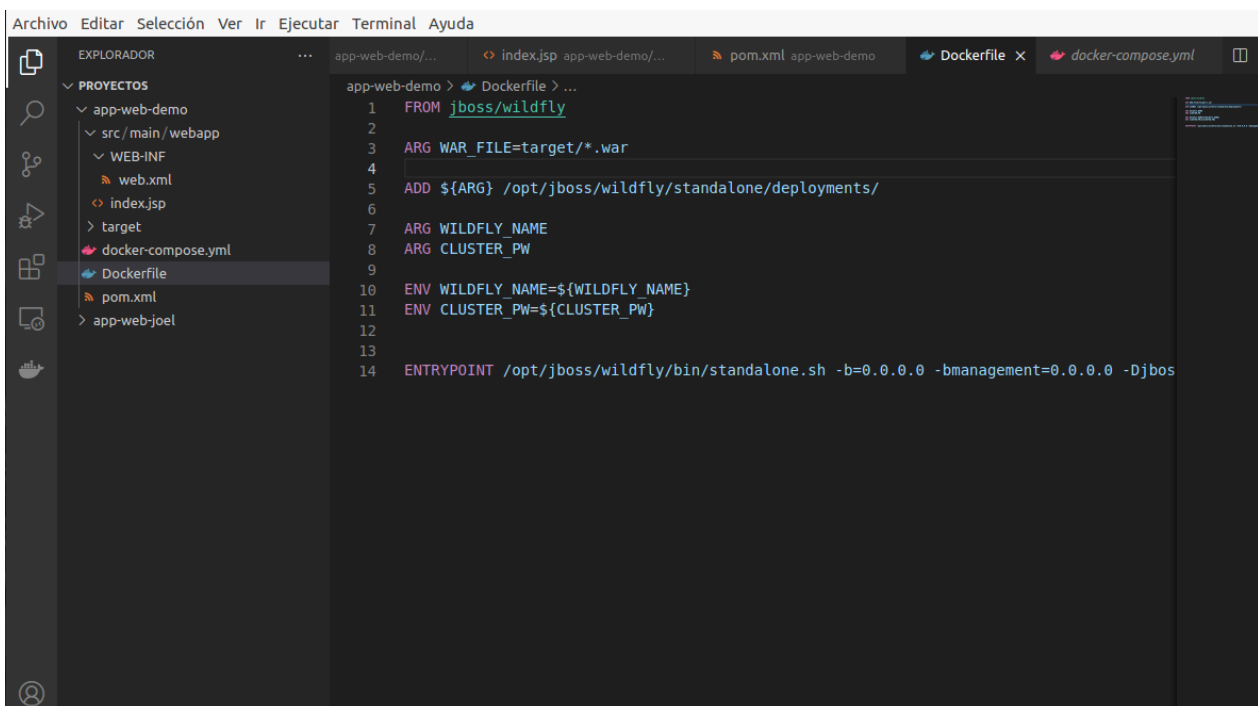
Seguidamente el fichero .yml (docker-compose.yml) para la construcción del cluster con el siguiente contenido

```
version: '3.5'
services:
wildfly1:
```

```

build:
context: .
args:
WILDFLY_NAME: wildfly_1
CLUSTER_PW: secret_password
image: wildfly_1
ports:
- 8080:8080
networks:
wildfly_network:
wildfly2:
build:
context: .
args:
WILDFLY_NAME: wildfly_2
CLUSTER_PW: secret_password
image: wildfly_2
ports:
- 8081:8080
networks:
wildfly_network:
networks:
wildfly_network:
ipam:
driver: default

```



```

Archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda
EXPLORADOR  app-web-demo/...  index.jsp app-web-demo/...  pom.xml app-web-demo  Dockerfile x  docker-compose.yml
PROYECTOS
  app-web-demo
    src/main/webapp
      WEB-INF
        web.xml
        index.jsp
      target
      docker-compose.yml
      Dockerfile
      pom.xml
      app-web-joel
app-web-demo > Dockerfile > ...
1  FROM jboss/wildfly
2
3  ARG WAR_FILE=target/*.war
4
5  ADD ${ARG} /opt/jboss/wildfly/standalone/deployments/
6
7  ARG WILDFLY_NAME
8  ARG CLUSTER_PW
9
10 ENV WILDFLY_NAME=${WILDFLY_NAME}
11 ENV CLUSTER_PW=${CLUSTER_PW}
12
13
14 ENTRYPOINT /opt/jboss/wildfly/bin/standalone.sh -b=0.0.0.0 -bmanagement=0.0.0.0 -Djboss

```

Observamos que existe: Dos servidores WILDFLY (1/2).

Distintos puertos para el arranque de cada uno.

CLUSTER_PW: clave para la construcción del cluster.

networks: wildfly_network. Subred que estamos construyendo.

ipam: Configuración de la subred.

Abriremos la carpeta Dockerfile y hacemos los siguiente

```
##ADD ${ARG} /opt/jboss/wildfly/standalone/deployments/  
ADD ${WAR_FILE} /opt/jboss/wildfly/standalone/deployments/
```

Posteriormente listaremos las imagenes con el comando

```
sudo docker-compose -a
```

Y luego las borramos todas con el siguiente comando

```
sudo docker rmi $(sudo docker images -a -q) -f
```

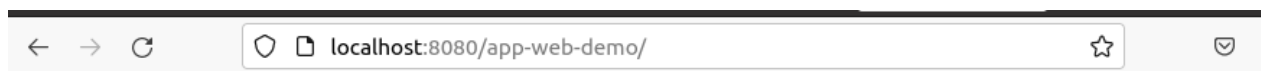
Ahora subiremos el war para desplegarlo en los dos puertos, con el comando

```
sudo docker-compose up --build
```

```
console listening on http://0.0.0.0:9990  
wildfly1_1 | 19:12:25,420 INFO [org.jboss.as.server] (Controller Boot Thread) WFLYSRV0212:  
Resuming server  
wildfly1_1 | 19:12:25,452 INFO [org.jboss.as] (Controller Boot Thread) WFLYSRV0025: WildFl  
y Full 25.0.0.Final (WildFly Core 17.0.1.Final) started in 49256ms - Started 524 of 759 serv  
ices (466 services are lazy, passive or on-demand)  
wildfly1_1 | 19:12:25,471 INFO [org.jboss.as] (Controller Boot Thread) WFLYSRV0060: Http m  
anagement interface listening on http://0.0.0.0:9990/management  
wildfly1_1 | 19:12:25,478 INFO [org.jboss.as] (Controller Boot Thread) WFLYSRV0051: Admin  
console listening on http://0.0.0.0:9990
```

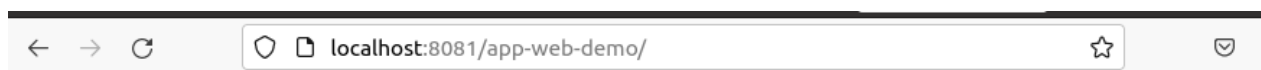
Ahora tendremos desplegada la aplicación accesible a través del puerto 8081

Puerto 8080



Este es el puerto: 8080

Puerto 8081



Este es el puerto: 8081

Enlace github: [enlace](#)