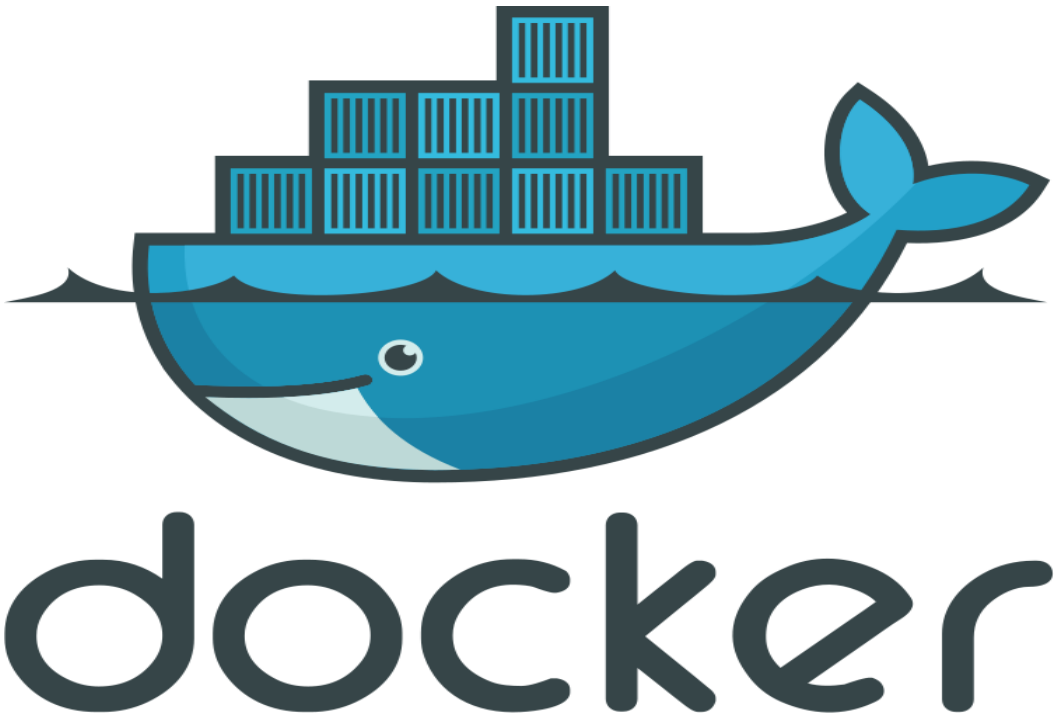
Oscar Rodríguez García, 2º DAW

Tarea 13: Clusterizacion de una app en Wildfly



Oscar Rodríguez García, 2º DAW

Índice

1. Introducción

La instalación convencional de **Jboss** requiere la instalación del nodo **master**, y la configuración de los nodos hijos o esclavos.

En cada instalación se debe evaluar la cantidad de instancias en modo Host que se desean utilizar y la cantidad de instancias en modo **Slave** que se levantaran en cada uno de los Hosts, por lo general esta información se obtiene a partir de las estimaciones de usuarios y llamadas concurrentes que se deberán soportar.

Comúnmente se configura un Host por servidor (físico o virtual) pero podría levantarse más de un Host en cada servidor, si el mismo cuenta con varias IPs.

Los pasos siguientes detallan la configuración a realizar para una infraestructura con dos Host (cada uno en un servidor distinto) con tres instancias/servers por HOST, todo agrupado en dos grupos.

Cada grupo será administrado de forma independiente, permitiendo desplegar las aplicaciones automáticamente en todos los servidores del grupo, así como configurar propiedades de cada uno de ellos en un único archivo.

2. Despliegue de un Cluster de JBOSS con Docker

Primero vamos a realizar algunos cambios en el fichero web.xml de nuestro proyecto:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
        id="WebApp_ID" version="3.0">

    <display-name>app-web-alumno</display-name>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

web.xml

src > main > webapp > WEB-INF > web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
        id="WebApp_ID" version="3.0">

    <display-name>app-web-oscar</display-name>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>

```

Luego realizaremos cambios en el fichero index.jsp

src > main > webapp > index.jsp > ? > html > body > h1

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
    <body>
        <h1>Hola soy Oscar!</h1>
        <p>Este es el puerto: ${pageContext.request.serverPort}</p>
    </body>
</html>

```

Posteriormente lanzamos el siguiente comando.

```
usuario@ubuntu-20:~/Escritorio/prueba/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.$ReflectU
tils$1 (file:/usr/share/maven/lib/guice.jar) to method java.lang.ClassLoader.defineCl
ass(java.lang.String,byte[],int,int,java.security.ProtectionDomain)
WARNING: Please consider reporting this to the maintainers of com.google.inject.inter
nal.cglib.core.$ReflectUtils$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective a
ccess operations
WARNING: All illegal access operations will be denied in a future release
[INFO] Scanning for projects...
[INFO]
[INFO] -------------------< es.system.alumno:app-web-demo >--------------------
```

Una vez construido el proyecto, puedes ver el resultado en ejecutando en modo local:

```
usuario@ubuntu-20:~/Escritorio/prueba/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.$ReflectU
tils$1 (file:/usr/share/maven/lib/guice.jar) to method java.lang.ClassLoader.defineCl
ass(java.lang.String,byte[],int,int,java.security.ProtectionDomain)
WARNING: Please consider reporting this to the maintainers of com.google.inject.inter
nal.cglib.core.$ReflectUtils$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective a
ccess operations
```

Comprobamos que la salida sea la correcta:

Ahora construimos el fichero Dockerfile

```
Dockerfile
1   FROM jboss/wildfly
2
3   ARG WAR_FILE=target/*.war
4   ##COPY ${JAR_FILE} app.jar
5
6   ADD ${ARG} /opt/jboss/wildfly/standalone/deployments/
7
8   ARG WILDFLY_NAME
9   ARG CLUSTER_PW
10
11  ENV WILDFLY_NAME=${WILDFLY_NAME}
12  ENV CLUSTER_PW=${CLUSTER_PW}
13
14  ENTRYPOINT /opt/jboss/wildfly/bin/standalone.sh -b=0.0.0.0 -bmanageme
15
```

Y después construimos el fichero .yml

docker-compose.yml

```yaml
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
```

Y por último realizamos el siguiente comandos para crear el cluster de dos nodos

```
usuario@ubuntu-20:~/Escritorio/prueba/app-web-demo$ sudo docker-compose up
Creating network "app-web-demo_wildfly_network" with the default driver
Building wildfly1
Sending build context to Docker daemon  10.24kB
Step 1/8 : FROM jboss/wildfly
 ---> 856694040847
```