

## **Módulo de Proyecto Integrado**

**Fco Javier Frías Serrano - 2º de ASIR**

**Rafael Miguel Cruz Álvarez - 2º de ASIR**

**Proyecto: Docker**

**Curso 2015/2016**

---

### **Índice de contenido**

1.- Introducción.....	2
2.- Objetivos y requisitos del proyecto.....	4
3.- Estudio previo.....	5
3.1.- Estado actual.....	5
3.2.- Estudio de soluciones existentes.....	6
4.- Plan de trabajo.....	6
5.- Diseño.....	7
5.1.- Diseño general.....	7
5.2.- Diseño detallado.....	9
6.- Implantación.....	12
7.- Recursos.....	42
7.1.- Herramientas hardware.....	42
7.2.- Herramientas software.....	42
7.3.- Personal.....	42
7.4.- Presupuesto.....	42
8.- Conclusiones.....	43
8.1.- Grado de consecución de objetivos.....	43
8.2.- Problemas encontrados.....	43
8.3.- Futuras mejoras.....	43
9.- Referencias / bibliografía.....	44

# 1.- Introducción

El instituto IES Gran Capitán le gustaría poder guardar los proyectos integrados de alumnos o grupos de alumnos de otros años, de forma que no le afecten las actualizaciones que se realicen en el servidor y que facilite de alguna manera la realización de copias de seguridad de los mismos.

El proyecto consiste inicialmente, en montar una máquina virtual en el servidor Júpiter del instituto IES Gran Capitán que contenga un servidor de contenedores Docker y cada contener es un proyecto integrado realizado años anteriores por otros alumnos o grupos de alumnos del centro.

De esta forma, facilitaremos la aplicación de actualizaciones y copias de seguridad sin riesgo a perder dichos proyectos integrados.

## ¿Qué es Docker?

Es una herramienta open-source que permite una virtualización ligera con la que es posible poder empaquetar entornos y aplicaciones para desplegarlo posteriormente en cualquier sistema que disponga de esta tecnología.

Puede definirse como un sistema que ejecuta máquinas virtuales ligeras, menos exigentes con los recursos de los equipos.

## ¿Por qué utilizar Docker y no Máquinas Virtuales?

La principal diferencia es que una máquina virtual necesita contener todo el Sistema Operativo mientras que un contenedor Docker aprovecha el Sistema Operativo sobre el cual se ejecuta, es decir, comparte el kernel del sistema operativo anfitrión e incluso parte de sus bibliotecas.

Esto le permite tener unas características de ligereza, portabilidad y autosuficiencia.

- Portabilidad, debido a que podemos desplegar un contenedor Docker en cualquier otro sistema (que soporte la tecnología Docker), ahorrándonos instalar un nuevo entorno para aquellas aplicaciones que normalmente usamos.
- Ligereza, ya que este sistema ahorra muchos recursos a la hora de desplegar un contenedor.
- Autosuficiencia, puesto que un contenedor de Docker no contiene todo un sistema completo, sino tan sólo aquellas librerías, archivos y configuraciones necesarias para desplegar las funcionalidades que contenga. Además el propio sistema Docker se encarga de la gestión del contenedor y de las aplicaciones que contenga.

### **Ventajas**

- Las instancias se arrancan en pocos segundos.
- Es fácil de automatizar e implementar en entornos de integración continua.
- Existen multitud de imágenes que se pueden descargar y modificar libremente.

### **Desventajas**

- Sólo puede usarse de forma nativa en Unix.
- Las imágenes sólo pueden estar basadas en versiones Linux modernas (*Kernel 3.8 mínimo*).
- Como es relativamente nuevo, puede haber errores de código entre versiones.

### **Solución híbrida**

En determinados escenarios puede tener muchas ventajas utilizar la tecnología de Docker junto a las máquinas virtuales. Un ejemplo de esto sería el desarrollo de una aplicación y utilizar ambos sistemas para probar el software desarrollado, corriendo varias pruebas para ver que la aplicación se está ejecutando correctamente y simulando el entorno de un cliente lo más idéntico posible.

## 2.- Objetivos y requisitos del proyecto

El principal objetivo es el ahorro de recursos al no necesitar máquinas virtuales para los distintos proyectos de otros años, pues gracias a los contenedores de Docker utilizaremos las librerías indispensables para ejercer sus funciones. Con esto liberamos una serie de recursos que podrán ser empleados por el instituto para otros fines.

Los requisitos para desarrollar la solución Docker serán:

- Una máquina virtual Linux de 64 bits y kernel superior a 3.10, en nuestro caso hemos usado un debian 8.6.
- Conocimientos teóricos y prácticos de Docker
- Proyectos de otros años con los contenedores previamente preparados.

**Nota 1:** El uso de un contenedor como máquina virtual Linux está totalmente desaconsejado, puesto que un contenedor quitará funciones a la máquina virtual servidora y Docker no podrá tener un correcto funcionamiento.

**Nota 2:** Se recomienda realizar las pruebas en un servidor Linux local para evitar errores en el servidor principal. Una vez el contenedor funcione correctamente, crear una imagen con Docker y utilizarla en el servidor en producción.

## 3.- Estudio previo

### 3.1.- Estado actual

Los proyectos inventario, moodle antiguo y labordeta se encuentran en el servidor Júpiter del instituto Gran Capitán, cada uno instalado en un contenedor distinto de proxmox.

- inventario (inventario.iesgrancapitan.org)
- moodle-old (moodle-old.iesgrancapitan.org)
- biblioinv: <http://biblioinv.iesgrancapitan.org/> ( contenedor 128, roy/makkay)
- glpi
- Web homenaje a Labordeta (labordeta.iesgrancapitan.org)

Jupiter: root@192.168.12.6

Para ver los contenedores activos: vzlist

(-a muestra los inactivos también)

```
root@jupiter:~# vzlist
CTID      NPROC STATUS   IP_ADDR      HOSTNAME
103        76 running  192.168.12.103 ubuntu1404.gcap.net
108        43 running  192.168.12.108 inventario.gcap.net
109        39 running  192.168.12.109 gestor-cocina.gcap.net
115        45 running  192.168.12.115 intranet.gcap.net
128        50 running  192.168.12.128 serviciosgc.gcap.net
129        52 running  192.168.12.129 serviciospr.gcap.net
143        36 running  192.168.12.143 labordeta.gcap.net
163        151 running  192.168.12.163 alfresco.gcap.net
111        - stopped 192.168.12.111 moodle-old.gcap.net
```

### 3.2.- Estudio de soluciones existentes

La solución principal es la de instalar Docker engine y crear distintos contenedores donde irán almacenados estos proyectos. Docker nos permitirá realizar imágenes de los contenedores con los servicios que queramos ya sea apache, mysql, etc. Lo bueno que tiene docker es que nos permitirá realizar actualizaciones de cada contenedor por separado, es decir si por ejemplo el proyecto inventario necesita que se actualice una nueva versión de mysql, solo actualizaremos su contenedor.

## 4.- Plan de trabajo

Planificación temporal de las fases de diseño general, diseño detallado e implantación.

<i>Fases del proyecto</i>	<i>Detalles de la fase</i>	<i>Tiempo estimado</i>
Investigación de Docker	Análisis de la información existente sobre la solución Docker para ser implantada en el centro de estudios.	Del 28 de Septiembre al 6 de Octubre
Pruebas de funcionamiento de Docker	Pruebas de funcionamiento de los proyectos en local con la instalación de Docker antes de proceder a su implementación en el servidor	Del 6 Octubre al 17 de Noviembre
Investigación del entorno de implantación	Análisis del entorno donde se pretende implantar Docker	Del 17 Noviembre al 24 Noviembre
Implantación en el servidor Júpiter	Puesta en funcionamiento de la solución Docker	Del 24 Noviembre al 12 Diciembre

## 5.- Diseño

### 5.1.- Diseño general

A continuación se muestran los datos de nuestro contenedor en el Servidor Júpiter.

Settings	
Key ▲	Value
cpus	1
disk	4
hostname	proyectodockers
ip_address	192.168.12.230
memory	512
nodename	jupiter
ostemplate	local:vztmpl/debian-8.0-x86_64.tar.gz
searchdomain	gcap.net
storage	local
swap	512
vmid	230

### Especificaciones del Servidor Júpiter del Instituto.

```
root@jupiter:~# lshw -C CPU
*-cpu
  description: CPU
  product: Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz
  vendor: Intel Corp.
  physical id: 4
  bus info: cpu@0
  version: Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz
  serial: To Be Filled By O.E.M.
  slot: LGA1155
  size: 3400MHz
  capacity: 3800MHz
  width: 64 bits
  clock: 100MHz
  capabilities: x86-64 fpu fpu_exception wp vme de pse tsc msr pae mce c
apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht
pbe syscall nx rdtscp constant_tsc arch_perfmon pebs bts rep_good xtopology
stop_tsc aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3
16 xtpr pdcm pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave av
ahf_lm ida arat epb pln pts dtherm tpr_shadow vnmi flexpriority ept vpid xsave
configuration: cores=4 enabledcores=1 threads=2
```

```

root@jupiter:~# uname -sr
Linux 2.6.32-45-pve
root@jupiter:~# lsb_release -idc
Distributor ID: Debian
Description:    Debian GNU/Linux 7.10 (wheezy)
Codename:       wheezy

root@jupiter:~# uname -sr
Linux 2.6.32-45-pve
root@jupiter:~# lsb_release -idc
Distributor ID: Debian
Description:    Debian GNU/Linux 7.10 (wheezy)
Codename:       wheezy

```

## Diferencia de tamaño entre Docker y Mvs.

El tamaño del disco duro que está usando actualmente el proyecto inventario en el servidor Júpiter es de 10GB en total mientras que si usamos un contenedor Docker, el tamaño de éste se vera reducido al tamaño total de la imagen que en este caso son 499.9 MB.

Tamaño Imagen inventario.

```

contenedor_inventario latest 1e9fc6acecd2 4 weeks ago
499.9 MB
docker@dockerdebian8:~$ _

```

Tamaño disco MV Inventario.

Processors	1
Memory	512MB
Swap	512MB
Disk size	10.00GB



## 5.2.- Diseño detallado

La instalación se realizará sobre un debian 8 de 64 bit y kernel 3.10 en modo texto, creado en el servidor Jupiter del IES Gran Capitán. Para poder acceder a él, lo haremos mediante el uso de ssh en la dirección [root@cpd.iesgrancapitan.org](mailto:root@cpd.iesgrancapitan.org) y el puerto del contenedor proxmox asignado.

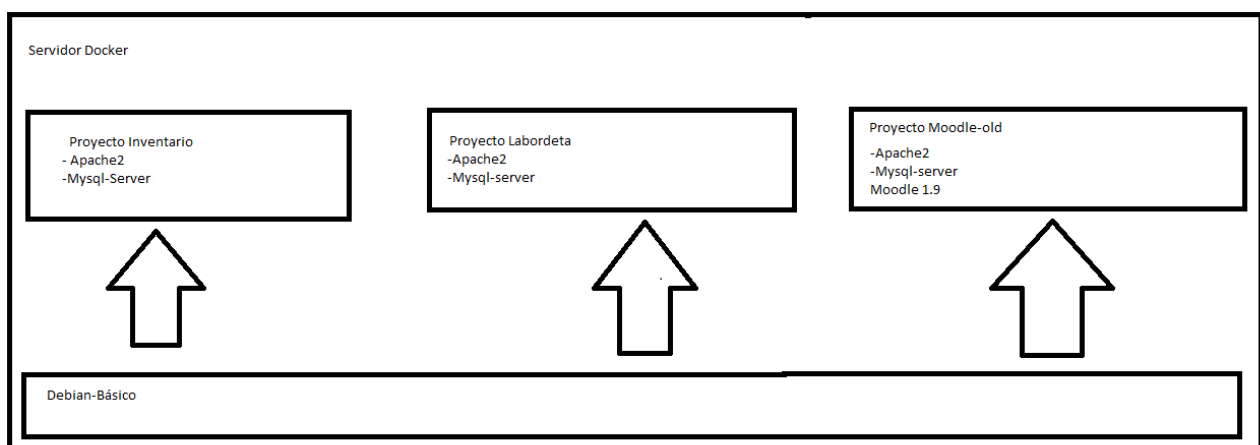
En nuestro debian, procedemos primeramente a la instalación de docker-engine, que sera el servicio que nos permitirá ejecutar y crear nuestros contenedores.

El instituto nos ha pedido que el proyecto se realice de la siguiente forma:

Primero creemos un contenedor con una imagen debian descargada del hub de docker (la cual viene sin ningún tipo de instalación básica) en la que debemos de instalar los componentes básicos, como puede ser los editores vi y nano, el ifconfig para poder ver las redes del contenedor, wget para poder realizar descargas, etc...

Una vez creado la imagen de éste debian a la que llamaremos debian\_básico, procedemos a la creación de los contenedores para los proyectos tal y como se muestran en la fase de Implantación.

Estructura del Servidor Docker:



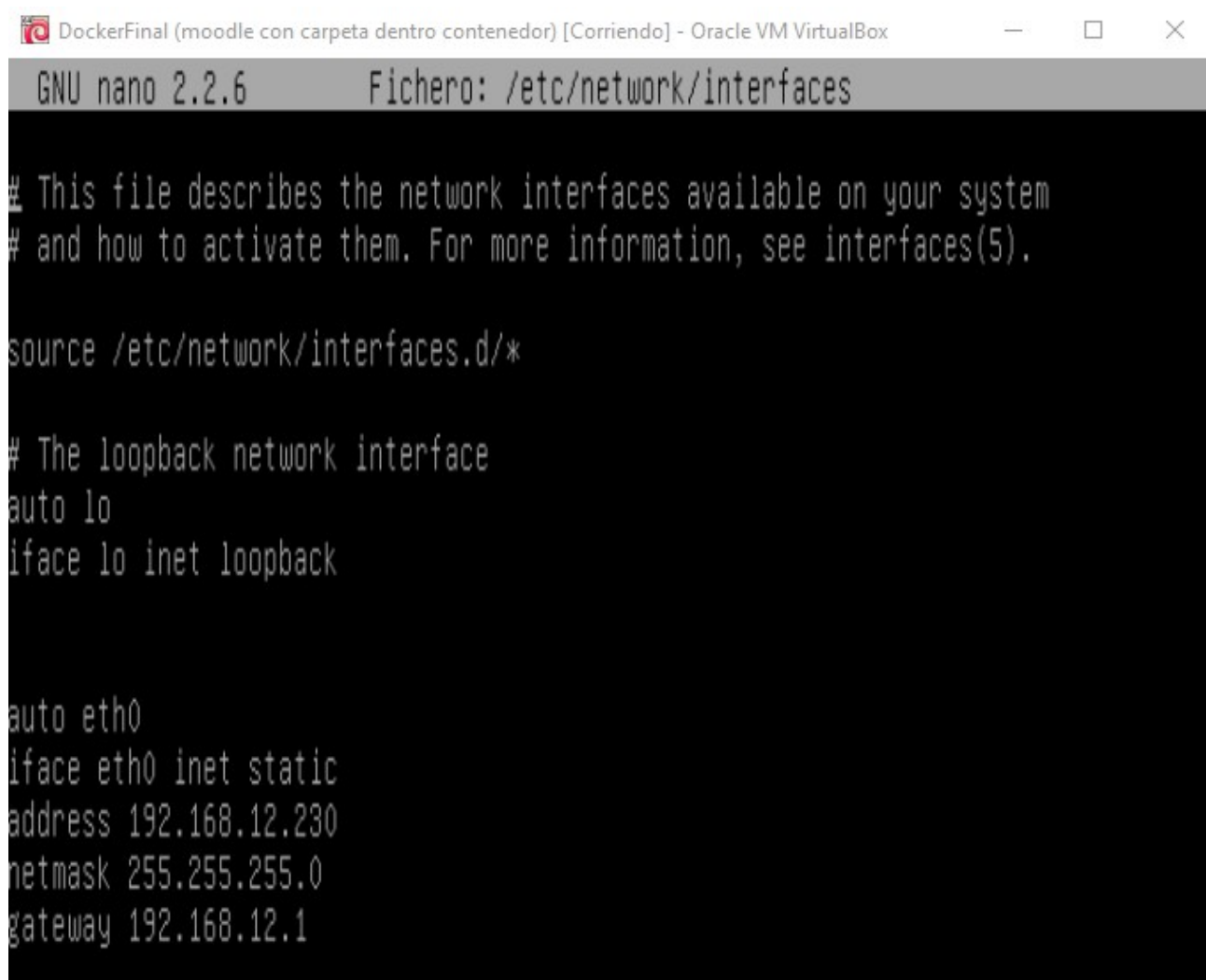
A continuación mostraremos como está configurado en el servidor Júpiter nuestro servidor debian.

Para acceder a él lo haremos mediante ssh como se ha indicado anteriormente con el usuario y pass siguiente

Acceso al contenedor desde el exterior: con el usuario dockers/dockers

**ssh dockers@cpd.iesgrancapitan.org:9230**

La configuración de red de nuestro servidor será la siguiente:



The screenshot shows a terminal window titled "DockerFinal (moodle con carpeta dentro contenedor) [Corriendo] - Oracle VM VirtualBox". The terminal is running GNU nano 2.2.6 and editing the file /etc/network/interfaces. The content of the file is as follows:

```
GNU nano 2.2.6      Fichero: /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.12.230
netmask 255.255.255.0
gateway 192.168.12.1
```

La configuración del resolv.conf es la siguiente.



```
DockerFinal (moodle con carpeta dentro contenedor) [Corriendo] - Oracle VM VirtualBox
GNU nano 2.2.6      Fichero: /etc/resolv.conf
# Generated by NetworkManager
domain gcap.net
search gcap.net
nameserver 80.58.61.250
```

Proxy inverso:

***inventario-docker.iesgrancapitan.org:80 -> redirige a 192.168.12.230:81***

***labordeta-docker.iesgrancapitan.org:80 -> redirige a 192.168.12.230:82***

***moodleold-docker.iesgrancapitan.org:80 -> redirige a 192.168.12.230:83***

## Configuración realizada por el centro para la redirección.

Proxy inverso (Nginx) : 192.168.12.106

Config en /etc/nginx/sites-available

En la config del site se especifica "proxy-pass" que es para no cambiar la URL.

Hay que indicar la IP del contenedor o MV.

Hemos copiado los sites antiguos de inventario, labordeta....

```
cp inventario inventario-docker
```

```
cp labordeta labordeta-docker
```

Para activarlos creamos enlace simbólico en sites-enabled

```
ln -s /etc/nginx/sites-available/inventario-docker /etc/nginx/sites-enabled/inventario-docker
```

Añadir entrada a DNS (en un servidor externo [www.iesgrancapitan.org](http://www.iesgrancapitan.org) de interdominios)

En /etc/bind:

inventario-docker IN CNAME informatica-gcap2.

## 6.- Implantación

### Instalación de Docker engine y prueba de funcionamiento y creación de contenedores.

En esta fase nos vamos a centrar en la instalación de Docker en Debian 8 Jessie y en su puesta a punto con una pequeña demostración de como crear los contenedores e imágenes.

Lo primero de todo será mostrar los pasos a seguir para la instalación de Docker.

Para instalar Docker, en mi caso lo he realizado sobre Debian 8.2.0 Jessie de 64bit (Muy importante que sea x64 ya que en x32 no funciona) y kernel mínimo 3.10.

Lo primero que hemos hecho ha sido realizar un **uname -m** para comprobar la arquitectura de Debian.

```
root@dockerins:~# uname -m
x86_64
root@dockerins:~# _
```

A continuación debemos de hacer un update y un upgrade del sistema.

```
root@dockerins:~# apt-get update && apt-get upgrade_
```

Una vez lo tenemos actualizado, procedemos a la creación de la clave GPG.

```
root@dockerins:~# apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADBF76221572C52609D
```

Y nos saldrá que ya se ha creado la key para Docker.

```
p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADBF76221572C52609D
gpg: solicitando clave 2C52609D de hkp servidor p80.pool.sks-keyservers.net
gpg: clave 2C52609D: clave pública "Docker Release Tool (releasedocker) <docker@docker.com>" importada
gpg: Cantidad total procesada: 1
gpg:          importadas: 1  (RSA: 1)
root@dockerins:~#
```

Ahora creamos el fichero docker.list en /etc/apt/source.list.d donde escribiremos el repositorio de Docker.



```
GNU nano 2.2.6  Fichero: /etc/apt/sources.list.d/docker.list
deb https://apt.dockerproject.org/repo debian-jessie main
```

Como este repositorio necesita https instalamos apt-transport-https y ca-certificates.

```
root@dockerins:~# apt-get install apt-transport-https ca-certificates_
```

Una vez hecho esto hacemos de nuevo un apt-get update.

Una vez hecho todos estos pasos previos, procedemos a la instalación de Docker.

Realizamos un apt-get install docker-engine.

```
root@dockerins:~# apt-get install docker-engine
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes extras:
  aufs-tools cgroupfs-mount git git-man libapparmor1 liberror-perl
  libnih-dbus1 libnih1 makedev mountall plymouth rsync
Paquetes sugeridos:
```

Ahora usando systemctl start docker lo iniciamos.

```
root@dockerins:~# systemctl start docker
root@dockerins:~# _
```

Y vemos que con systemctl status docker está activado.

```
root@dockerins:~# systemctl status docker
• docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled)
   Active: active (running) since mié 2016-10-26 17:50:36 CEST; 1min 46s ago
     Docs: https://docs.docker.com
    Main PID: 3072 (dockerd)
     CGroup: /system.slice/docker.service
             └─3072 /usr/bin/dockerd -H fd://
               └─3076 docker-containerd -l unix:///var/run/docker/libcontainerd
oct 26 17:50:36 dockerins dockerd[3072]: time="2016-10-26T17:50:36.88195039"
oct 26 17:50:36 dockerins dockerd[3072]: time="2016-10-26T17:50:36.88204780"
oct 26 17:50:36 dockerins dockerd[3072]: time="2016-10-26T17:50:36.88206419"
```

Para comprobar que todo se ha instalado correctamente realizaré la prueba con el contenedor de Hola Mundo.

```
https://docs.docker.com/engine/userguide/
root@dockerins:~# docker run hello-world_
<
```

Aquí vemos como se ha creado el contenedor.

```
status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
```

A continuación indicaremos los comandos necesarios para crear imágenes, descargarnos las imágenes, hacer correr los contenedores, etc..

Para ello lo que voy a hacer es crear un contenedor que contenga un Debian con un servidor Apache.

Primer paso seria descargarnos la imagen de debian, para ello usamos el comando **docker pull [nombre de la imagen]**

```
root@dockerins:~# docker pull debian
Using default tag: latest
latest: Pulling from library/debian
43c265008fae: Pulling fs layer
```

Para ver que imágenes tenemos usamos el comando **docker images**.

```
root@dockerins:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
debian               latest             7b0a06c805e8       4 days ago
123 MB
root@dockerins:~# _
```

Una vez descargado, procedemos a la creación del contenedor con el comando **docker run -i -t [nombre de la imagen] /bin/bash**

**-i y -t lo usamos para el modo interactivo y se nos asigne un tty.** Y así poder entrar dentro del contenedor.

```
root@dockerins:~# docker run -i -t debian /bin/bash
root@5f9d3508216f:/# _
```

Es muy importante que siempre que creamos un contenedor realicemos un apt-get update  
Como lo que queremos instalar es apache2, realizamos un apt-get install apache2.

Una vez instalamos es importante reiniciar apache2, por lo que hacemos un /etc/init.d/apache2 restart.

Cuando hayamos reiniciado el servicio apache2, debemos de presionar Ctrl+p+q para que se salga del contenedor pero sin pararlo ya que a continuación se va a proceder a crear una imagen de dicho contenedor.



Si ejecutamos el comando **docker ps** podemos ver como el contenedor está levantado.

```
dockerinstalacion [Corriendo] - Oracle VM VirtualBox
root@dockerins:~# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED
STATUS        PORTS      NAMES
5f9d3508216f   debian    "/bin/bash"             7 minutes ago
Up 7 minutes   cocky_hugle
root@dockerins:~# _
```

Volviendo al contenedor, como lo tenemos abierto, todavía se puede perder la información. Para que esto no suceda tenemos que guardar el contenedor. Para guardar el contenedor le tenemos que asignar un nombre distinto, ya que no se pueden sobrescribir contenedores.

Para ello usamos **docker commit [nombre del contenedor] [nombre que queremos poner]**

```
root@dockerins:~# docker commit cocky_hugle apache
sha256:ff7d3ef337c08cb183cd6b59ff887ce140d9c1ea3af7ee5c78efcea023dcf3f3
root@dockerins:~# _
```

Ahora si hacemos un docker images, podemos ver como se ha creado la imagen de apache.

```
root@dockerins:~# docker images
REPOSITORY      TAG          IMAGE ID          CREATED
SIZE
apache          latest      ff7d3ef337c0      56 seconds ago
192.9 MB
debian          latest      7b0a06c805e8      5 days ago
123 MB
root@dockerins:~# _
```

Esto nos permitirá usarla siempre que queramos.

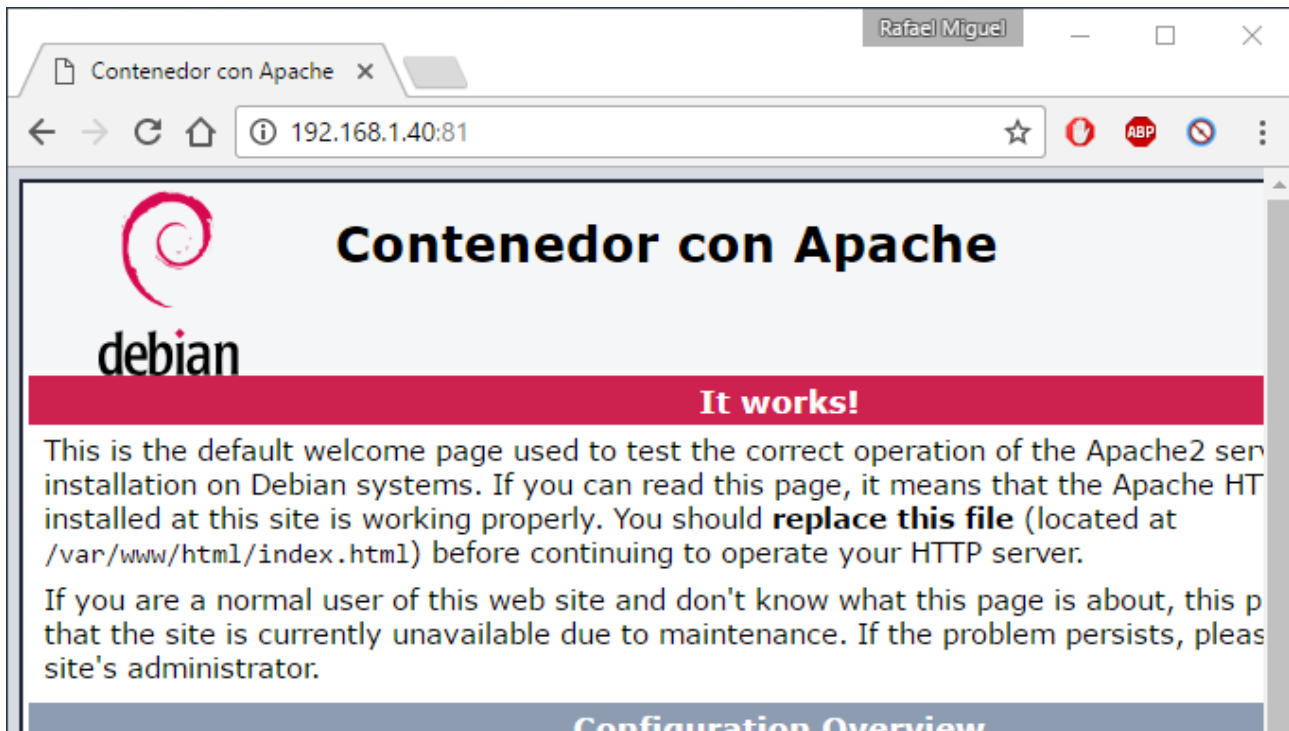
A continuación y para terminar vamos a hacer que el sitio de apache podamos verlo desde nuestro anfitrión, para ello usamos el comando **docker run -d -p [puerto del contenedor]:[puerto del servidor docker] [nombre de la imagen] /usr/sbin/apache2ctl -D FOREGROUND**.

```
root@dockerins:~# docker run -d -p 81:80 apache /usr/sbin/apache2ctl -D FOREGROUND
eef1e520e494330c57e73154a742f40f4fe6eb7d8cab6177e290b86b65bf7f0
root@dockerins:~# _
```

Y si hacemos un `docker ps` podemos ver el contenedor nuevo con la redirección de puertos.

```
root@dockerins:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS            PORTS              NAMES
eef1e520e49        apache             "/usr/sbin/apache2ctl"   About a minute
ago               Up About a minute  0.0.0.0:81->80/tcp      loving_goodall
root@dockerins:~# _
```

Solo nos queda hacer la comprobación desde el anfitrión, para ello escribimos la ip del servidor de docker:81.



**Y con esto se concluye la prueba de funcionamiento de docker. A continuación se explicarán las instalaciones de los proyectos en diferentes contenedores.**

## **Proyecto Inventario.**

### **Instalación y configuración del Inventario.**

Para realizar la instalación de este proyecto, necesitaremos primero un contenedor que tenga instalado Debian en su version 8. En él realizaremos una serie de instalaciones y configuraciones para poner en funcionamiento el inventario.

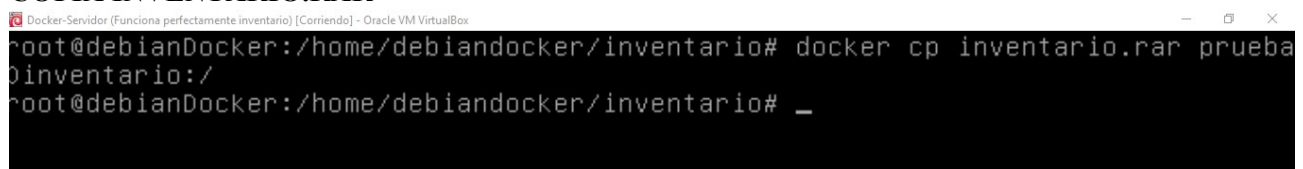
Como vemos, creamos el contenedor llamado prueba0inventario con una imagen Debian descargada anteriormente.



```
Docker-Servidor [Funciona perfectamente inventario] [Corriendo] - Oracle VM VirtualBox
root@debianDocker:/home/debiandocker# docker run -i -t --name prueba0inventario
debian /bin/bash_
```

Una vez creado el contenedor, procedemos a copiar desde nuestro servidor Debian el inventario.rar y la base de datos. Para ello desde el servidor realizamos el siguiente comando desde la ruta donde tenemos el proyecto y su bd → `docker cp inventario.rar (nombrecontenedor):(rutadondeguardar)`. Con esto copiamos lo que necesitamos dentro de nuestro contenedor, en nuestro caso la guardamos en el raíz.

### **COPIA INVENTARIO.RAR**



```
Docker-Servidor [Funciona perfectamente inventario] [Corriendo] - Oracle VM VirtualBox
root@debianDocker:/home/debiandocker/inventario# docker cp inventario.rar prueba0inventario:/
root@debianDocker:/home/debiandocker/inventario# _
```

### **COPIA DE LA BASE DE DATOS.**



```
Docker-Servidor [Funciona perfectamente inventario] [Corriendo] - Oracle VM VirtualBox
root@debianDocker:/home/debiandocker# docker cp bdd_inventario prueba0inventario:/
root@debianDocker:/home/debiandocker# _
```

Una vez hechas las copias de los ficheros necesarios para hacer funcionar el inventario, procedemos a la instalación de los servicios que vamos a necesitar.

Primero debemos de realizar un apt-get update para llevar a cabo una actualización de los paquetes.

```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@15e66f856253:~# apt-get update
Get:1 http://security.debian.org jessie/updates InRelease [63.1 kB]
Get:2 http://security.debian.org jessie/updates/main amd64 Packages [408 kB]
Ign http://deb.debian.org jessie InRelease
Get:3 http://deb.debian.org jessie-updates InRelease [145 kB]
Get:4 http://deb.debian.org jessie Release.gpg [2373 B]
Get:5 http://deb.debian.org jessie Release [148 kB]
Get:6 http://deb.debian.org jessie-updates/main amd64 Packages [17.6 kB]
Get:7 http://deb.debian.org jessie/main amd64 Packages [9064 kB]
71% [7 Packages 6185 kB/9064 kB 68%] 101 kB/s 28s^
Fetched 9848 kB in 10s (938 kB/s)
Reading package lists... Done
root@15e66f856253:~# _
```

A continuación procedemos con la instalación de apache2 para poder poner el funcionamiento del sitio. Para ello escribimos apt-get install apache2.

```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@15e66f856253:~# apt-get install apache2_
```

Ahora procedemos a la instalación de mysql-server para nuestra base de datos. Realizamos un apt-get install mysql-server.

```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@15e66f856253:~# apt-get install mysql-server_
```

Nos pedirá que introduzcamos una password para root.

```
While not mandatory, it is highly recommended that you set a password for the
MySQL administrative "root" user.

If this field is left blank, the password will not be changed.

New password for the MySQL "root" user:
```

A continuación instalamos los paquetes de php necesarios, apt-get install php5 php-pear php5-mysql php5-gd.



```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@15e66f856253:~# apt-get install php5 php-pear php5-mysql php5-gd_
```

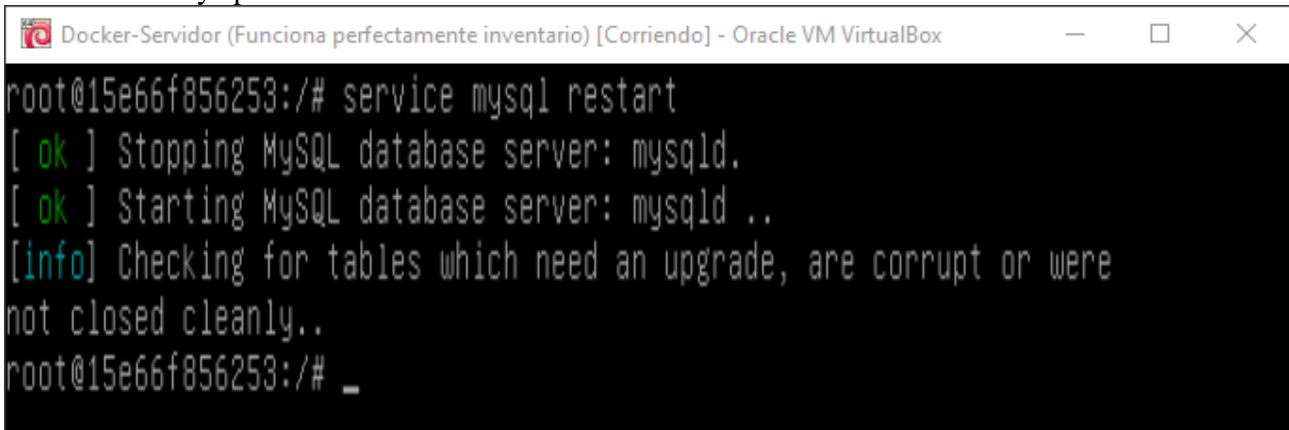
Y con esto, ya tenemos todo lo necesario instalado. Ahora toca realizar las configuraciones necesarios para su funcionamiento.

Empezaremos configurando mysql. Para ello es necesario reiniciar primero mysql ya que si no lo hacemos nos saldrá el siguiente error.




```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@efa4500c974d:/# mysql -u root -p
Enter password:
ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/var/run
/mysqld/mysqld.sock' (2)
root@efa4500c974d:/# _
```

Reiniciamos mysql-server.



```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@15e66f856253:/# service mysql restart
[ ok ] Stopping MySQL database server: mysqld.
[ ok ] Starting MySQL database server: mysqld ..
[info] Checking for tables which need an upgrade, are corrupt or were
not closed cleanly..
root@15e66f856253:/# _
```

Ejecutamos el comando `mysql -u root -p` e indicamos la contraseña que hayamos puesto en la instalación, y nos saldrá una especie de terminal que nos permitirá usando los comandos necesarios la creación de la base de datos, modificarla, ver tablas, etc...



```
root@15e66f856253:/# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 37
Server version: 5.5.53-0+deb8u1 (Debian)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

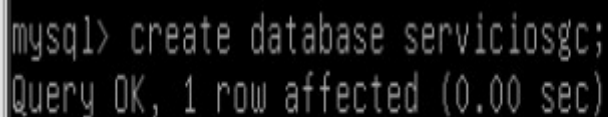
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> _
```

Lo primero que vamos a hacer es crear la base de datos (que se tiene que llamar igual que como viene en el fichero `bbdd_inventario.sql` en este caso es `serviciosgc`), para crearla usamos:

**CREATE DATABASE serviciosgc;**



```
mysql> create database serviciosgc;
Query OK, 1 row affected (0.00 sec)
```

A continuación creamos el usuario tal y como viene en el fichero config.php del proyecto inventario.

**CREATE USER (nombreusuario)@ip;**

```
mysql> create user serviciosgc@localhost;  
Query OK, 0 rows affected (0.00 sec)
```

Ahora le asignamos una pass a ese usuario (Tambien disponible en el config.php de inventario)

**SET PASSWORD FOR (usuario)@ip= PASSWORD (“contraseña”);**

```
mysql> set password for serviciosgc@localhost= PASSWORD ("Ser_0950");  
Query OK, 0 rows affected (0.00 sec)
```

Por último damos permisos.

**GRANT ALL PRIVILEGES ON (nombrebasededatos).\* TO (usuario)@localhost IDENTIFIED BY ‘contraseña’;**

```
mysql> grant all privileges on serviciosgc.* to serviciosgc@localhost identified  
by 'Ser_0950';  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> _
```

**FLUSH PRIVILEGES;**

```
mysql> flush privileges;  
Query OK, 0 rows affected (0.00 sec)
```



**Nos salimos con quit.**

```
mysql> quit
Bye
root@15e66f856253:/# _
```

Con esto acabamos con la creación de la base de datos.

Ahora procedemos con la instalación del sitio.

Primero copiamos inventario.rar en /var/www/html y lo descomprimos.

```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@15e66f856253:/home# cp inventario.rar /var/www/html
root@15e66f856253:/home# ls /var/www/html/
index.html  inventario.rar
root@15e66f856253:/home# _
```

Como el debian que instalamos viene vacío, debemos de instalar unrar-free para poder descomprimir el fichero inventario.rar. Una vez descomprimido, podemos borrar el fichero inventario.rar

```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@15e66f856253:/var/www/html# apt-get install unrar-free_
```

```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@15e66f856253:/var/www/html# unrar inventario.rar _
```

```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@15e66f856253:/var/www/html# ls
index.html inventario inventario.rar
root@15e66f856253:/var/www/html# rm -r inventario.rar _
```

Una vez descomprido, hacemos un `chown www-data:www-data /var/www/html/inventario/* -R` para darle esos permisos a todo lo de dentro de la carpeta. Y hacemos un `usermod -a -G www-data root`.

```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@15e66f856253:/var/www/html# chown www-data:www-data /var/www/html/inventari
o/* -R
```

```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@15e66f856253:/var/www/html/inventario# usermod -a -G www-data root
root@15e66f856253:/var/www/html/inventario# _
```

Vemos que todos los ficheros de dentro de inventario tiene permisos `www-data`.

```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
total 248
-rw-r--r-- 1 www-data www-data 949 Nov 16 09:58 ArtBaja.php
drwxr-xr-x 5 www-data www-data 4096 Nov 16 10:02 CodBarras
-rw-r--r-- 1 www-data www-data 5228 Nov 16 09:58 articulo.php
drwxr-xr-x 2 www-data www-data 4096 Nov 16 10:02 autocompletar
-rw-r--r-- 1 www-data www-data 1978 Nov 16 09:58 borrar.php
-rw-r--r-- 1 www-data www-data 4778 Nov 16 09:58 browser_detection.php
-rw-r--r-- 1 www-data www-data 1285 Nov 16 09:58 browser_plataforma.php
drwxr-xr-x 2 www-data www-data 4096 Nov 16 09:59 calendario
-rw-r--r-- 1 www-data www-data 2802 Nov 16 09:58 categoria.php
-rw-r--r-- 1 www-data www-data 672 Nov 16 09:58 cerrar.php
```

Ya que tenemos los permisos cambiados, nos vamos a /etc/apache2/sites-available y creamos un sitio nuevo, lo hemos llamado inventario.conf y realizamos su configuración, para crearlo, lo que hacemos es hacer una copia del sitio 000-default.conf, ya que es el que viene configurado por defecto y lo único que hay que hacer es cambiar la ruta del documento como mostraremos a continuación (Importante: Cambiar el puerto del sitio 000-default ya que nuestro sitio ira por el puerto 80).

```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@15e66f856253:/etc/apache2/sites-available# cp 000-default.conf inventario.conf
root@15e66f856253:/etc/apache2/sites-available# ls
000-default.conf  default-ssl.conf  inventario.conf
root@15e66f856253:/etc/apache2/sites-available# _
```

Cambiamos como hemos indicado antes el puerto de 000-default.conf y el documentroot de inventario.conf.

```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
GNU nano 2.2.6      File: inventario.conf      Modified
<VirtualHost *:80>
    #ServerName www.example.com

    # ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html/inventario_

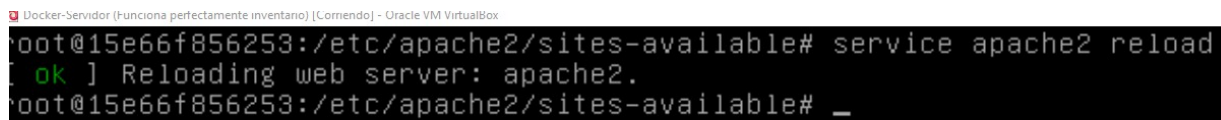
    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
```

Una vez configurado, levantamos el sitio con a2ensite inventario.conf y reinicamos apache2.

```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@15e66f856253:/etc/apache2/sites-available# a2ensite inventario.conf
Enabling site inventario.
To activate the new configuration, you need to run:
    service apache2 reload
root@15e66f856253:/etc/apache2/sites-available# _
```

Hacemos un reload de apache2.



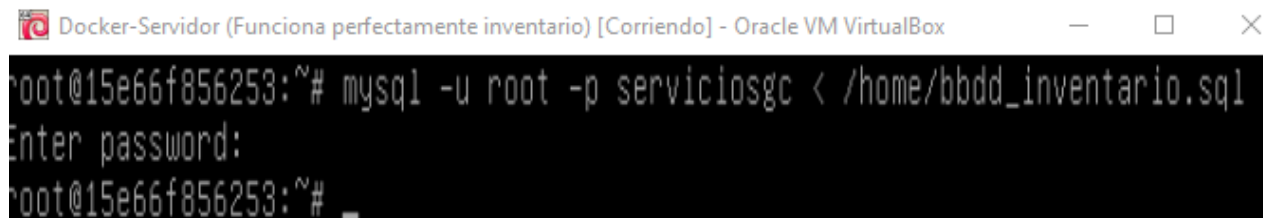
```
root@15e66f856253:/etc/apache2/sites-available# service apache2 reload
[ ok ] Reloading web server: apache2.
root@15e66f856253:/etc/apache2/sites-available# _
```

Con esto ya tenemos nuestro sitio web en funcionamiento.

Ahora necesitamos importar la base de datos, para ello realizamos el siguiente comando:

`mysql -u root -p (nombrebasededatos → serviciosgc) < /home/bbdd_inventario.sql`

Y con esto se importa la base de datos.



```
root@15e66f856253:~# mysql -u root -p serviciosgc < /home/bbdd_inventario.sql
Enter password:
root@15e66f856253:~# _
```

Comprobamos que la base de datos se ha importado correctamente. Para ello hacemos un `mysql -u serviciosgc -p` y metemos la contraseña.

```
root@15e66f856253:~# mysql -u serviciosgc -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 39
Server version: 5.5.53-0+deb8u1 (Debian)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> _
```

Una vez dentro de mysql, debemos de activar la base de datos, para ello usamos el comando use y el nombre de la base de datos, en este caso seria use serviciosgc; .

```
mysql> use serviciosgc;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> _
```

Para mostrar las bases de datos, usamos show databases; .

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| serviciosgc |
+-----+
2 rows in set (0.00 sec)
```

Vemos como nuestro usuario serviciosgc solo tiene acceso a la base de datos serviciosgc.  
Para mostrar las columnas hacemos lo siguiente.

```
mysql> show tables from serviciosgc;
+-----+
| Tables_in_serviciosgc |
+-----+
| categorias |
| ele_ubi |
| elementos |
| estado_incidencias |
| familias |
| incidencias |
| procedencias |
| tipo_incidencias |
| ubicaciones |
| unidades_organizativas |
| usuarios |
+-----+
11 rows in set (0.00 sec)
```

Para poder interactuar con una columna, lo único que debemos de realizar son consultas, por ejemplo, para saber el admin del inventario, usamos lo siguiente.

```
mysql> select * from usuarios;
```

id	nombre	apellidos	usuario	clave	correo	estado	id_unidad	perfil

Y nos mostraría la tabla con los datos.

Con esto terminamos con la configuraciones de apache2 para nuestro sitio web y con mysql-server para nuestra base de datos.

Ahora viene lo más importante que es la instalación y configuración de supervisord (programa que nos permitirá mediante el uso de script el funcionamiento (ejecución) de varios servicios dentro de nuestro contenedor.)

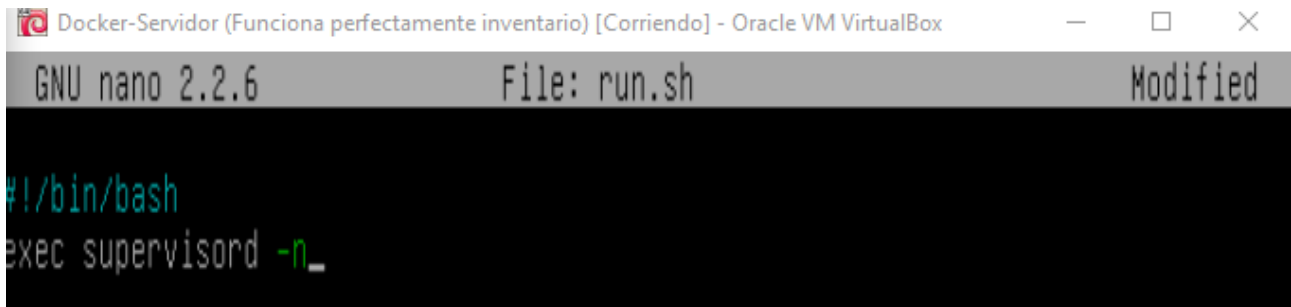
Hacemos un apt-get install supervisor para instalarlo en nuestro contenedor.

```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@15e66f856253:~# apt-get install supervisor
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libpython-stdlib libpython2.7-minimal libpython2.7-stdlib python
  python-meld3 python-minimal python-pkg-resources python2.7 python2.7-minimal
Suggested packages:
  python-doc python-tk python-distribute python-distribute-doc python2.7-doc
  binutils binfmt-support
```

Una vez instalado, procedemos a la creación de los scripts.

Para ello nos vamos a la ruta / y creamos un script llamado run.sh que será el encargado de poner en funcionamiento los servicios en el arranque del contenedor.

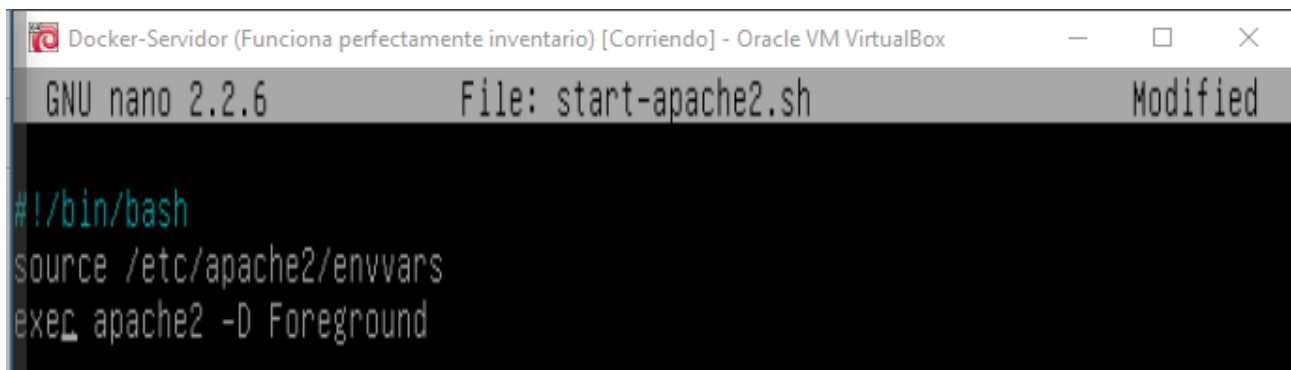
## IMAGEN RUN.SH

A screenshot of a terminal window titled "Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox". The terminal shows the GNU nano 2.2.6 editor editing the file run.sh. The content of the script is: 

```
#!/bin/bash
exec supervisord -n_
```

Una vez creado run.sh, creamos otros dos script, uno que usaremos para ejecutar Apache y otro para ejecutar mysql, esto también lo creamos en el directorio raiz.

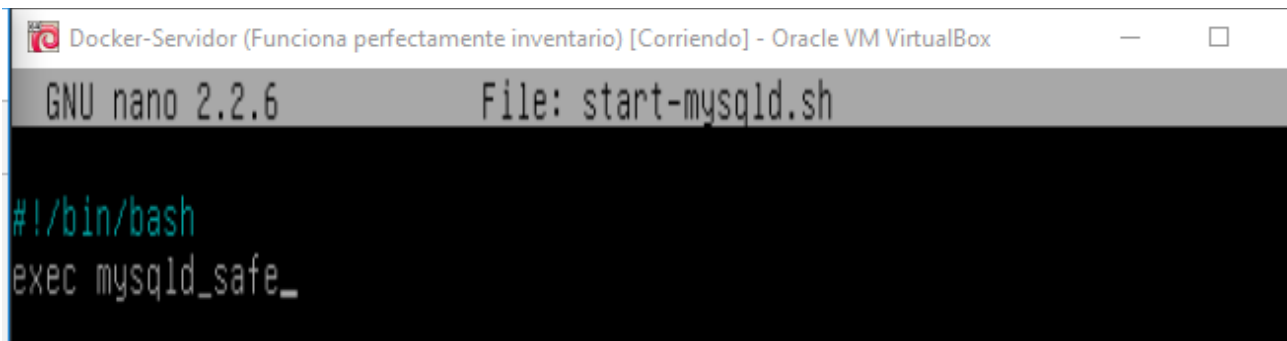
## IMAGEN APACHE2.

A screenshot of a terminal window titled "Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox". The terminal shows the GNU nano 2.2.6 editor editing the file start-apache2.sh. The content of the script is: 

```
#!/bin/bash
source /etc/apache2/envvars
exec apache2 -D Foreground
```




## IMAGEN MYSQL.



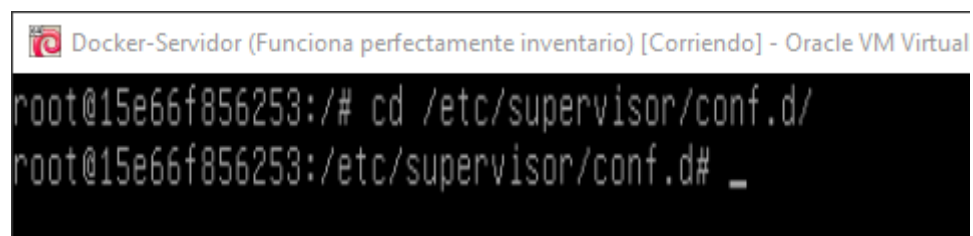
```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
GNU nano 2.2.6 File: start-mysqld.sh
#!/bin/bash
exec mysqld_safe_
```

Una vez creados los scripts le damos permisos con `chmod 777 nombrescript`.



```
root@15e66f856253:/# chmod 777 run.sh start-apache2.sh start-mysqld.sh
root@15e66f856253:/# ls -l run.sh start-apache2.sh start-mysqld.sh
-rwxrwxrwx 1 root root 32 Nov 16 10:47 run.sh
-rwxrwxrwx 1 root root 67 Nov 16 10:48 start-apache2.sh
-rwxrwxrwx 1 root root 29 Nov 16 10:49 start-mysqld.sh
root@15e66f856253:/# _
```

Ahora que tenemos creados los scripts nos vamos a la configuración de supervisor y en `/etc/supervisor/conf.d` creamos dos ficheros que serán los encargados de poner en funcionamiento nuestro servicios a través de supervisor.



```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM Virtual
root@15e66f856253:/# cd /etc/supervisor/conf.d/
root@15e66f856253:/etc/supervisor/conf.d# _
```

## IMAGEN APACHE2

Docker-Servidor (Funciona todo inventario) [Corriendo] - Oracle VM VirtualBox

```
GNU nano 2.2.6 File: supervisord-apache2.conf

[program:apache2]
command=/start-apache2.sh
numprocs=1
autostart=true
autorestart=true
```

## IMAGEN MYSQL.

Docker-Servidor (Funciona todo inventario) [Corriendo] - Oracle VM VirtualBox

```
GNU nano 2.2.6 File: supervisord-mysqld.conf

[program:mysqld]
command=/start-mysqld.sh
numprocs=1
autostart=true
autorestart=true
_
```

Con esto terminamos la instalación y configuración del contenedor.

Nos salimos con `ctrl+p+q` para dejar activo el contenedor y procedemos a crear una imagen del mismo usando el comando `docker commit (nombrecontenedor) (nombreimagen)`.

Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox

```
root@debianDocker:/home/debiandocker# docker commit prueba0inventario prueba0inv
entario_
```

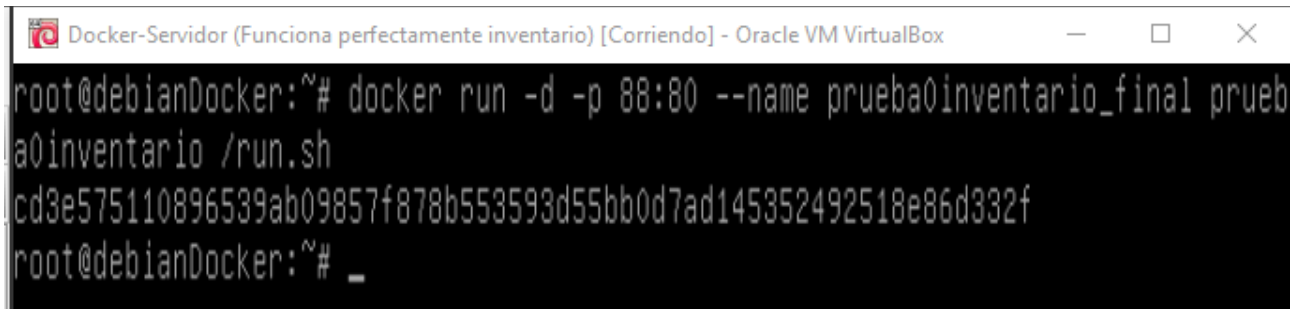
Hacemos un `docker images`, para comprobar que se ha creado la imagen de nuestro contenedor.

Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox

```
root@debianDocker:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
prueba0inventario   latest             105029a7e903       26 seconds ago
499.5 MB
```

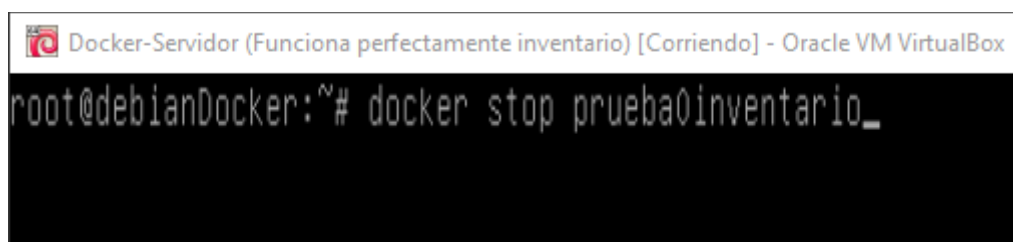
Hecho esto creamos el contenedor con esa imagen y lo redirigimos el puerto 88 de nuestro contenedor al 80 del servidor para poder visualizar nuestra web.

`docker run -d -p 88:80 --name(nombre que le queremos poner) (nombreimagen) /run.sh` (script creado antes).



```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@debianDocker:~# docker run -d -p 88:80 --name prueba0inventario_final prueba0inventario /run.sh
cd3e575110896539ab09857f878b553593d55bb0d7ad145352492518e86d332f
root@debianDocker:~# _
```

Ya podemos parar el contenedor que hemos usado para crear esta imagen, para ello hacemos un `docker stop prueba0inventario`.



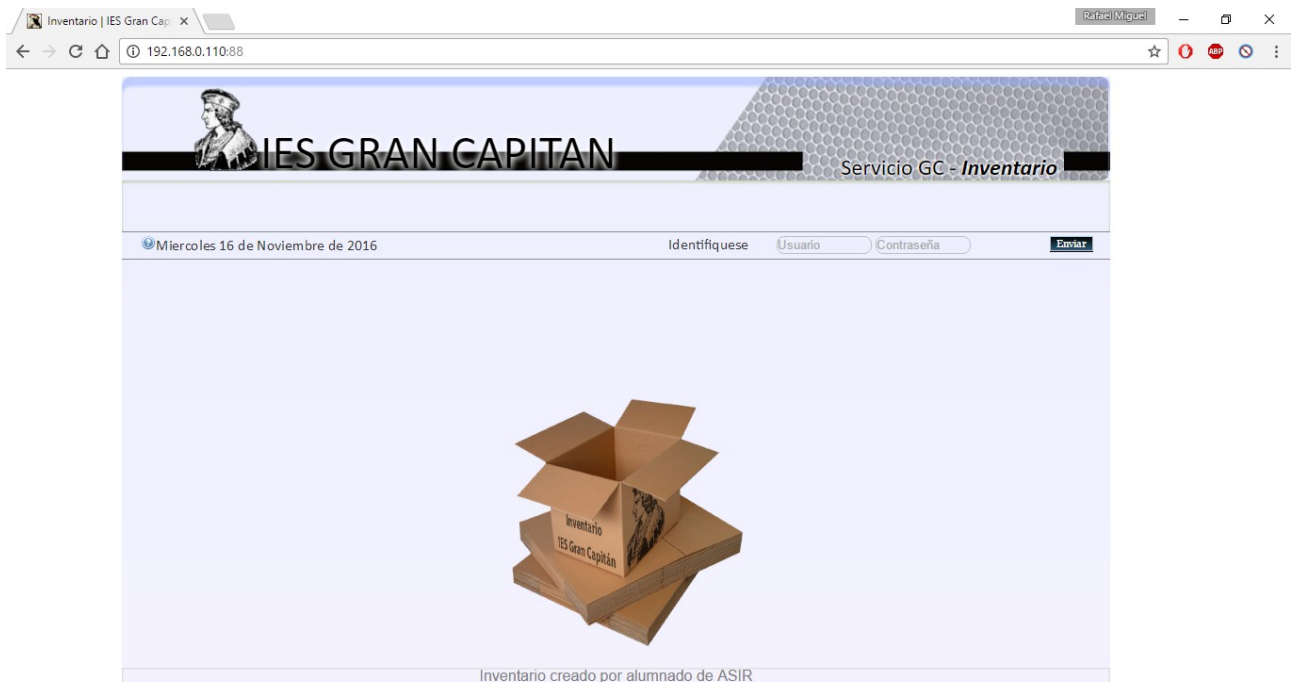
```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@debianDocker:~# docker stop prueba0inventario_
```

Con `docker ps` vemos el contenedor que esta activo.

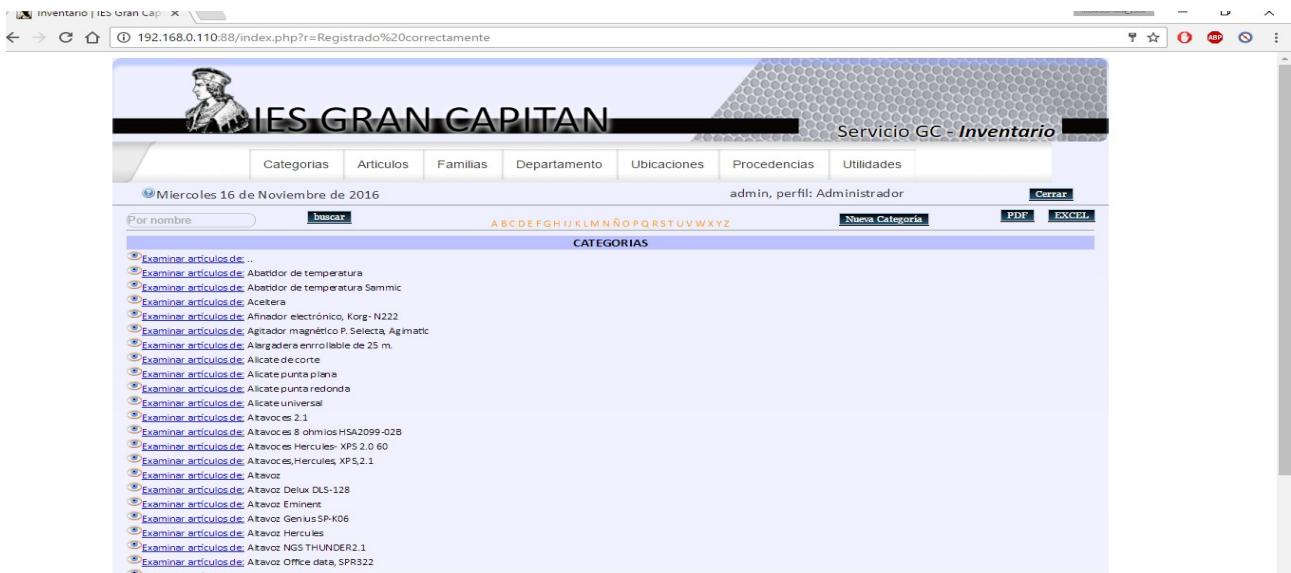


```
Docker-Servidor (Funciona perfectamente inventario) [Corriendo] - Oracle VM VirtualBox
root@debianDocker:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS            PORTS              NAMES
cd3e57511089        prueba0inventario  "/run.sh"          About a minute ago
Up About a minute   0.0.0.0:88->80/tcp  prueba0inventario_final
root@debianDocker:~# _
```

Y ahora nos vamos a un navegador escribimos la ip del servidor:88 y ahí está nuestro inventario en funcionamiento, para probarlo, ingresamos con el usuario admin que esta en nuestra base de datos y podemos ver como funciona.



Ingresamos con el usuario admin y su contraseña.



Y vemos que funciona correctamente.

## Proyecto Labordeta

La instalación del proyecto Labordeta se realizará siguiendo el mismo procedimiento que con la instalación del inventario, ya que los servicios que usaremos serán apache2 y la instalación de mysql-server para la base de datos.

## Proyecto Moodle-old.

Para realizar la instalación del moodle-old, primero debemos de instalar moodle en su versión 1.9, para ello hemos seguido el siguiente tutorial.

[https://docs.moodle.org/all/es/Instalando\\_Moodle\\_en\\_distribuciones\\_basadas\\_en\\_Debian](https://docs.moodle.org/all/es/Instalando_Moodle_en_distribuciones_basadas_en_Debian)

Una vez instalado moodle, lo que debemos de hacer es sustituir la carpeta moodledata del moodle-old por la carpeta moodledata que hemos creado durante la instalación y también hemos sustituido la base de datos creada por la del moodle-old (para ver como se sustituye en el apartado de mysql del proyecto inventario se explica).

## Instalación en el servidor.

Una vez tenemos probados los proyectos en local, toca instalarlo en el servidor Júpiter.

Para ello lo primero que debemos de hacer es conectarnos por ssh a [docker@cpd.iesgrancapitan.org](mailto:docker@cpd.iesgrancapitan.org):9230. Una vez tenemos instalado docker-engine en nuestro Debian, procedemos a pasar por ssh las copias de las imágenes de los proyectos que hemos realizado en local.

Para realizar estas copias debemos de usar el siguiente comando:

**docker save [imagen] > [archivo].tar (esto se hará en el servidor local para exportar las imágenes ya creadas)**

**docker load < [archivo].tar (Esto lo haremos en el servidor Júpiter una vez nos hayamos pasado el tar por ssh lo extraemos).**

Una vez tengamos extraídas las imágenes de nuestros contenedores en nuestro servidor debian en Júpiter, procedemos a crear los contenedores con dichas imágenes.

Para ello lo que hacemos es usar el comando `docker run -d -p 81:80 --name proyecto_inventario (nombre_imagen que hemos extraído) /run.sh`

Haremos lo mismo cambiando de puerto (82 para labordeta y 83 para moodle) para crear los contenedores de estos proyectos.

Una vez tengamos los contenedores con nuestros tres proyectos, procedemos a la redirección de puertos para que se puedan ver desde fuera de la red que se explica en el siguiente punto.

## Redirección de los proyectos.

Una vez instalados todos los proyectos en el servidor toca redireccionarlos para que se puedan ver desde fuera de la red.

Para ello nos vamos a `/etc/apache2/sites-available` y creamos un sitio para cada proyecto.

```
root@dockerdebian8:/etc/apache2/sites-available# ls
000-default.conf  default-ssl.conf  inventario.conf  labordeta.conf
root@dockerdebian8:/etc/apache2/sites-available# _
```

Tal y como se muestra en la siguiente imagen configuramos el sitio por puertos.



```
<VirtualHost 192.168.12.230:81>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.

    ServerName inventario-docker.iesgrancapitan.org
```

```
# For most configuration files from conf-available/, which are
# enabled or disabled at a global level, it is possible to
# include a line for only one particular virtual host. For example the
# following line enables the CGI configuration for this host only
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

Como vemos, hemos creado el sitio por puertos y lo unico que debemos de poner es ServerName inventario-docker.iesgrancapitan.org.

Haremos lo mismo con el de labordeta.

```
<VirtualHost 192.168.12.230:82>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.

    ServerName labordeta-docker.iesgrancapitan.org
```

```
# For most configuration files from conf-available/, which are
# enabled or disabled at a global level, it is possible to
# include a line for only one particular virtual host. For example the
# following line enables the CGI configuration for this host only
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

Y con ésto podemos acceder desde fuera de la red sin tener que poner la ip y el puerto correspondiente.

Una vez hecho esto solo queda la configuración en el DNS del servidor para el alojamiento externo.

### **Configuración realizada por el centro para la redirección.**

Proxy inverso (Nginx) : 192.168.12.106

Config en /etc/nginx/sites-available

El la config del site se especifica "proxy-pass" que es para no cambiar la URL.

Hay que indicar la IP del contenedor o MV.

Hemos copiado los sites antiguos de inventario, labordeta....

```
cp inventario inventario-docker
```

```
cp labordeta labordeta-docker
```

Para activarlos creamos enlace simbólico en sites-enabled

```
ln -s /etc/nginx/sites-available/inventario-docker /etc/nginx/sites-enabled/inventario-docker
```

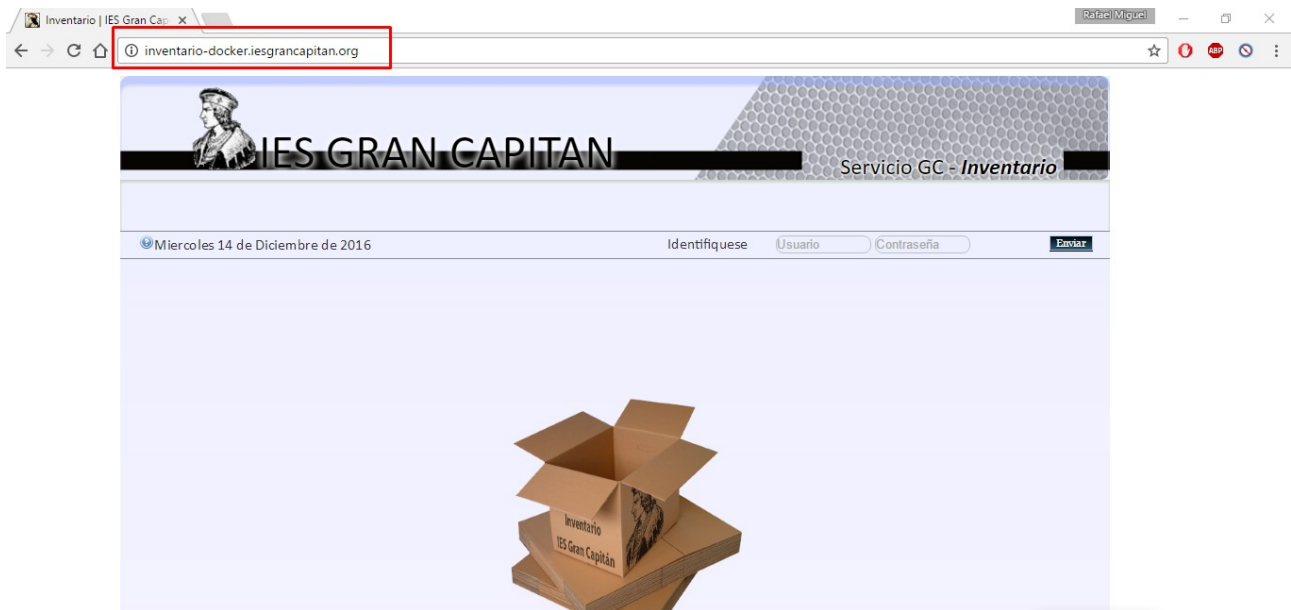
Añadir entrada a DNS (en un servidor externo [www.iesgrancapitan.org](http://www.iesgrancapitan.org) de interdominios)

En /etc/bind:

```
inventario-docker IN CNAME informatica-gcap2.
```



Inventario cuya dirección es 192.168.12.230:81 se podrá acceder a través de inventario-docker.iesgrancapitan.org.



Labordeta cuya dirección es 192.168.12.230:82 se podrá acceder a través de labordeta-docker.iesgrancapitan.org.



Con esto ya tenemos todo instalado y listo para poder ver los proyectos desde fuera de la red.

## 7.- Recursos

### 7.1.- Herramientas hardware

La máquina debe ser capaz de soportar una máquina virtual con Docker instalado. Para ello necesitara unos recursos como buena memoria RAM y procesador para atender las peticiones que se realicen a la misma con bastante fluidez. Además necesitará un espacio de almacenamiento adicional, puesto que a parte del espacio mínimo que necesita el Sistema Operativo instalado en la máquina virtual, necesitará espacio para contener los proyectos de años anteriores.

### 7.2.- Herramientas software

- Máquina virtual Debian 8 64bits y kernel 3.10 o superior como servidor instalado en el servidor Júpiter del instituto IES Gran Capitán.
- Instalación e implementación de la solución de Docker y sus contenedores a los proyectos integrados.

### 7.3.- Personal

El proyecto de integración de Docker en el servidor de Júpiter, será realizando por dos técnicos:

- Rafael Miguel Cruz Álvarez
- Francisco Javier Frías Serrano

### 7.4.- Presupuesto

	<b>Coste por hora</b>
<b>Mano de obra</b>	300 Euros/técnico
<b>Implantación</b>	800 Euros
<b>Coste total</b>	1400 Euros

## 8.- Conclusiones

### 8.1.- Grado de consecución de objetivos

Para cada objetivo, grado de consecución (totalmente terminado, parcialmente realizado, no abordado, no

implementado...) y justificación en caso de que no esté totalmente terminado.

- Instalación Debian 8 en Servidor Júpiter → Totalmente Terminado.
- Instalación de Docker-Engine en Debian 8 → Totalmente Terminado.
- Creación de Imagen con un Debian\_Basico como uso para crear contenedor → Totalmente Terminado.
- Creación de los contenedores para instalar los proyectos → Totalmente Terminado.
- Proyecto Inventario instalado en su contenedor → Totalmente Terminado.
- Proyecto Labordeta instalado en su contenedor → Totalmente Terminado.
- Proyecto Moodle-old instalado en su contenedor → No terminado Totalmente.
- Redirección de los proyectos para acceder desde el exterior → Totalmente Terminado.

### 8.2.- Problemas encontrados

El primer problema que hemos tenido ha sido a la hora de intentar arrancar dos servicios en el contenedor de cada proyecto, es decir iniciar por ejemplo el servicio apache y mysql que solo permitía iniciar uno solo. Para resolverlo tras mucha búsqueda encontramos que usando Supervisor y tras su configuración nos permitía ejecutar ambos servicios (Mirar en el apartado 6 como se ha realizado ésta configuración).

Otro problema que nos ha surgido ha sido en el apartado de redireccionar los proyectos para que se vieran desde fuera de la red.

### 8.3.- Futuras mejoras

Añadir phpmyadmin para poder configurar las bases de datos desde interfaz web y no desde la consola de comandos.

Crear un script que permita arrancar los contenedores al iniciar el servidor.

Hacer que la redirecciones de los documento de moodle-old funcionen correctamente.

## **9.- Referencias / bibliografía**

Todos los libros, páginas web y documentos en general empleados para el desarrollo.

[https://es.wikipedia.org/wiki/Docker\\_\(software\)](https://es.wikipedia.org/wiki/Docker_(software))

<http://www.campusmvp.es/recursos/post/Que-es-Docker.aspx>

<https://www.docker.com/>

<https://github.com/brunocascio/docker-espanol>

<http://www.muylinux.com/2016/04/19/tutorial-docker>

<https://www.adictosaltrabajo.com/tutoriales/docker-for-dummies/>

<https://www.digitalocean.com/community/tutorials/docker-explicado-como-crear-contenedores-de-docker-corriendo-en-memcached-es>