Método.1. Uso de imágenes personalizadas

1.Actualizamos el servidor

Desde el servidor ejecutamos apt update, y éste actualizará el servidor. También realizamos la instalación de nano con el comando apt install nano, esta aplicación nos permitirá editar ficheros de texto con facilidad.

```
root@srv1:~# apt update
```

Figura 54: *comando de actualización en servidor.*

```
root@srv1:~/proyecto# apt install nano
```

Figura 55: *comando de instalación de aplicación nano.*

2. Directorio para el proyecto

Generamos una carpeta en el interior de este equipo con el comando mkdir proyecto, donde guardaremos los archivos y ficheros correspondientes además de realizar las correspondientes operaciones. Accederemos a este mismo con el comando cd proyecto.

```
root@srv1:~# mkdir proyecto
root@srv1:~# cd proyecto/
root@srv1:~/proyecto#
```

Figura 56: *creación de carpeta proyecto.*

3. Instalación de docker

Entonces continuamos con el comando de instalación de docker, apt install docker docker-compose docker.io

```
root@srv1:~/proyecto# apt install docker docker-compose docker.io
```

Figura 57: *comando de instalación de aplicación docker*

4. Ejecutamos contenedor Portainer

Por si fuera necesario, instalaremos el contenedor Portainer. Este nos ofrece una manera de visualizar los contenedores, imágenes y más detalles de la ejecución de docker.

Para ello implementaremos el siguiente comando, con el cual logramos la ejecución de Portainer en el puerto 9000.

docker run -d -p 8100:9000 --name=portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:latest

```
root@srv1:~/proyecto# docker run -d -p 8100:9000 --name=portainer --restart=always -v /var/run
/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:latest
```

Figura 58: *comando de creación y ejecución container Portainer*

5. Ejecución de docker-compose.yml

En el interior de Github se encuentra un archivo docker-compose.yml, este archivo será necesario para la ejecución de los contenedores del proyecto. Este documento nos importará las imágenes del proyecto subidas a docker Hub.

```
version: '3.1'

services:

  wordpress:
    image: alberto0505/wordpress-tienda:1.0
    restart: always
    ports:
      - 8080:80
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: exampleuser
      WORDPRESS_DB_PASSWORD: examplepass
      WORDPRESS_DB_NAME: exampledb
      WORDPRESS_CONFIG_EXTRA: |
        define( 'WP_HOME', 'http://{TU IP}:8080' );
        define( 'WP_SITEURL', 'http://{TU IP}:8080' );

    volumes:
      - wordpress:/var/www/html

  db:
    image: alberto0505/db-tienda:1.0
    restart: always
    environment:
      MYSQL_DATABASE: exampledb
      MYSQL_USER: exampleuser
      MYSQL_PASSWORD: examplepass
      MYSQL_RANDOM_ROOT_PASSWORD: '1'
    volumes:
      - db:/var/lib/mysql

volumes:
  wordpress:
  db:
```

Figura 59: *documento docker-compose.yml método 1*

Pero antes de continuar será necesario realizar una pequeña modificación, en los apartados en los que aparece “{TU IP}” es necesario cambiarlos por la ip del equipo o servidor en el cual realizaremos las operaciones. Accediendo a este documento con el comando nano docker-compose.yml nos pondrá a editarlo. Introducimos los cambios correspondientes, y lo guardamos pulsando Ctrl + O.

Con el uso del comando docker-compose -f {nombre del fichero} up dentro del repositorio en el que se encuentre el documento.

```
root@srv1:~/proyecto# docker-compose -f docker-compose-propio.yml up -d
Creating network "proyecto_default" with the default driver
Creating volume "proyecto_wordpress" with default driver
Creating volume "proyecto_db" with default driver
Creating proyecto_wordpress_1 ... done
Creating proyecto_db_1        ... done
```

Figura 60: *ejecución del docker-compose.yml*

Éste nos generará el proyecto al que se podrá acceder con el enlace {servidor o equipo local}:8080/ dando por finalizado este procedimiento.