



TAREA 05

**IMPLANTACIÓN DE
SISTEMAS SEGUROS DE
DESPLEGADO DE
SOFTWARE**

PUESTA EN PRODUCCIÓN SEGURA

ALBA MOREJÓN GARCÍA

2024/2025

Ciberseguridad en Entornos de las Tecnologías de la Información

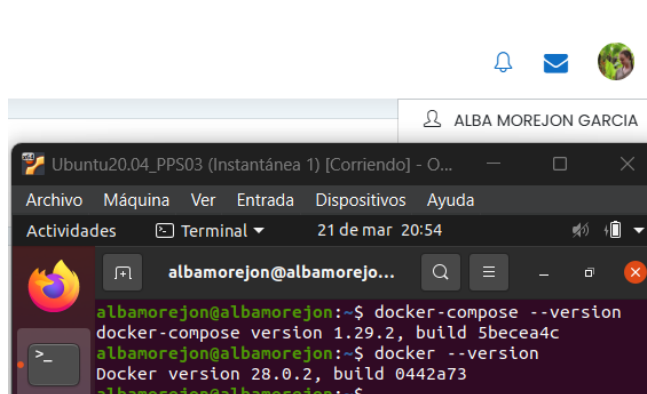
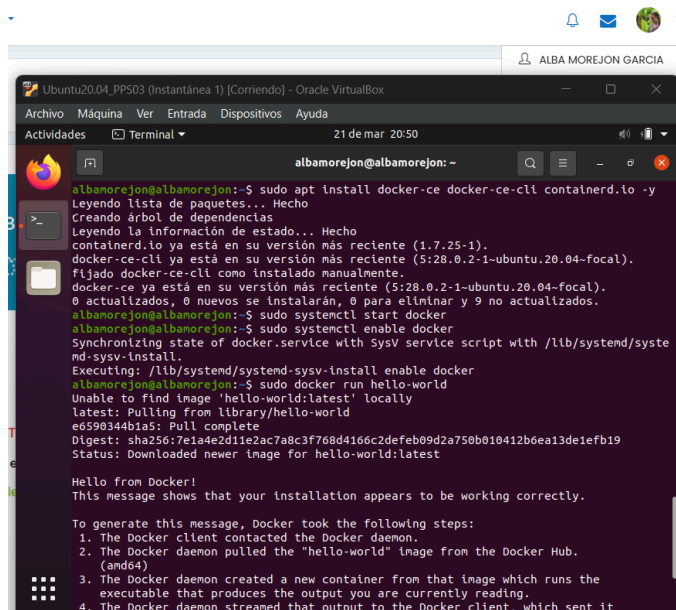
Caso práctico

Docker Logo, representa una ballena azul con contenedores al lado del nombre **dotCloud, Inc.**

Docker Logo (Apache License 2.0) Julián se ha unido a un grupo de trabajo para la creación de un aplicativo web para entornos web, sabe que sus compañeros están usando Docker y aún tiene cierto temor al uso de esta tecnología por desconocimiento por lo que va repasando cada paso que da.

Apartado 1: Manejo básico de contenedores con Docker

1- En las unidades 2 y 3 ya se ha utilizado Docker así que ya debes tenerlo instalado, en caso contrario debes instalarlo. Puedes instalarlo en tu equipo o crear una máquina virtual e instalarlo en la misma (en ese caso te recomendamos utilizar una máquina Ubuntu porque va a ser muy sencillo).



2- Rellena la siguiente tabla explicando que hacen las siguientes líneas

Comando/línea de comandos	Descripción detallada (si hay opciones se deben explicar)
<code>docker pull mysql:5.7.28</code>	Se utiliza para descargar una imagen específica de MySQL desde el repositorio de Docker a tu máquina local. <ul style="list-style-type: none">• <code>docker pull</code>, descarga la imagen desde un repositorio de Docker (en este caso desde Docker Hub)• <code>mysql:5.7.28</code>, especifica la imagen de MySQL y la versión a descargar
<code>docker images</code>	Devuelve una lista de las imágenes de Docker que estén almacenadas en la máquina local <ul style="list-style-type: none">• <code>docker</code>, comando que interactúa con el motor de Docker• <code>images</code>, lista las imágenes disponibles en la máquina
<code>docker run --rm --user mysql -d -p 33060:3306 --name mysql-db1 -e MYSQL_ROOT_PASSWORD=secret -v ./mysql:/var/lib/mysql mysql:5.7.28</code> (NOTA: si el equipo es windows se debe cambiar <code>./mysql</code> por <code>.\mysql</code>)	Inicia un contenedor de Docker que ejecuta la instancia de MySQL versión 5.7.28, se realiza en segundo plano, con el usuario mysql (configurando su contraseña) utilizando los puertos 33060:3306 y volúmenes para persistir los datos y permitir el acceso a MySQL desde el host <ul style="list-style-type: none">• <code>docker run</code>, inicia un nuevo contenedor a partir de una imagen Docker• <code>--rm</code>, elimina el contenedor cuando se detiene• <code>--user mysql</code>, ejecuta el contenedor con el usuario mysql• <code>-d</code> ejecuta el contenedor en segundo plano (modo detach)• <code>-p 33060:3306</code>, mapea el puerto 33060 del host al puerto 3306 del contenedor (permite acceder a MySQL en el puerto 3360 del host)• <code>--name mysql-db1</code>, da nombre al contenedor• <code>-e MYSQL_ROOT_PASSWORD=secret</code>, establece esa variable con el valor "secret" (configurando la contraseña del usuario root de MySQL)• <code>-v ./mysql:/var/lib/mysql</code>, monta el directorio <code>./mysql</code> del host en el directorio <code>/var/lib/mysql</code> del contenedor (permite que los datos se almacenen en el host)

	<p>asegurando su persistencia).</p> <ul style="list-style-type: none"> mysql:5-7-28, especifica la imagen de MySQL la versión a utilizar
docker ps -a	<p>Se utiliza para listar todos los contenedores de Docker en la máquina incluyendo los detenidos, mostrando su estado y los detalles de cada uno.</p> <ul style="list-style-type: none"> docker ps, lista los contenedores que están actualmente en ejecución, mostrando información de los contenedores: ID, imagen, nombre, estado y puertos -a, extiende la funcionalidad del comando para incluir todos los contenedores, los detenidos, en pausa o no iniciados también.
docker logs mysql-db1	<p>Muestra los registros (logs) de un contenedor específico de Docker, permitiendo revisar la salida estándar y los mensajes de error generados.</p> <ul style="list-style-type: none"> docker logs, muestra los registros de un contenedor de Docker, la salida estándar y la salida de error. mysql-db1, especifica el nombre del contenedor cuyos registros se quieren mostrar
docker exec -it mysql-db1 /bin/sh	<p>Ejecuta un comando dentro de un contenedor en ejecución, en este caso abre una sesión de shell dentro del contenedor mysql-db1.</p> <ul style="list-style-type: none"> docker exec, se ejecuta comando dentro del contenedor en ejecución -i, mantiene la entrada abierta, permitiendo la interacción con el contenedor -t, asigna una terminal pseudo-TTY, hace que la sesión sea interactiva y haga de máquina real. mysql-db1, nombre del contenedor en el que se quiere ejecutar el comando /bin/sh, comando que se ejecutará en el contenedor, en este caso se abre una shell permitiendo que interactúen con el sistema y ejecutar los comandos en el servidor
docker stop mysql-db1	<p>Sirve para detener el contenedor Docker llamado mysql-db1</p> <ul style="list-style-type: none"> docker stop, detiene un contenedor en ejecución enviando una señal SIGTERM y después SIGKILL mysql-db1, nombre del contenedor en el que se quiere detener
docker start mysql-db1	<p>Sirve para iniciar el contenedor Docker llamado mysql-db1</p> <ul style="list-style-type: none"> docker start, inicia el contenedor mysql-db1, nombre del contenedor en el que se quiere detener
docker rm -f mysql-db1	<p>Se utiliza para eliminar contenedores Docker (el contenedor debe haber sido detenido previamente) de forma forzada</p> <ul style="list-style-type: none"> docker rm, elimina uno o varios contenedores de Docker. -f, fuerza la eliminación del contenedor incluso si está en ejecución mysql-db1, nombre del contenedor en el que se quiere eliminar

3- Realizar pruebas con Jenkins utilizando contenedores.

Jenkins es una herramienta de integración continua y entrega continua (CI/CD) de código abierto, escrita en Java, que permite automatizar el desarrollo, prueba y despliegue de software. Visita su página oficial para entenderlo mejor: <https://www.jenkins.io/>

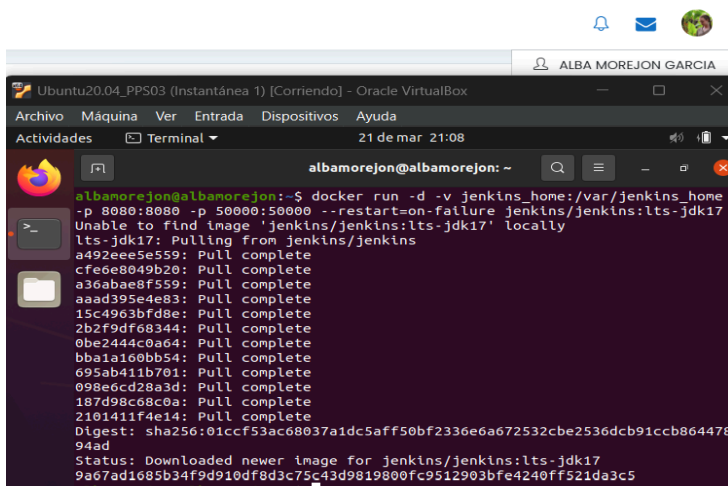
Incluye pantallazos de las pruebas que se piden a continuación, incluyendo las de la consola donde se lanzan las líneas de comandos docker y del navegador con la aplicación.

3.1- Lanzar el contenedor de Jenkins (<https://github.com/jenkinsci/docker/blob/master/README.md>).

Indica la línea de comandos para realizar esta operación.

```
docker run -d -v jenkins_home:/var/jenkins_home -p 8080:8080 -p 50000:50000 --restart=on-failure jenkins/jenkins:lts-jdk17
```

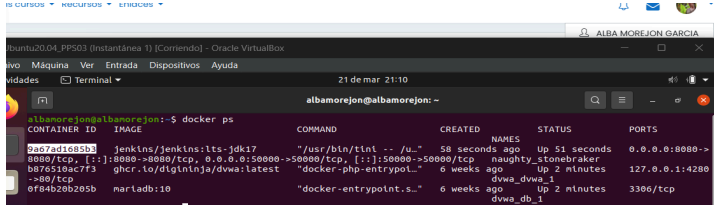
Este comando inicia un contenedor de Jenkins mapeando los puertos 8080 y 50000, y configura el contenedor para que se reinicie automáticamente en caso de fallo, asegura que los datos de Jenkins se mantengan persistentes y que el contenedor se ejecute en segundo plano



```
albamorejon@albamorejon:~$ docker run -d -v jenkins_home:/var/jenkins_home
-p 8080:8080 -p 50000:50000 --restart-on-failure jenkins/jenkins:lts-jdk17
Unable to find image 'jenkins/jenkins:lts-jdk17' locally
lts-jdk17: Pulling from jenkins/jenkins
a492eee5e559: Pull complete
cfe6e8049b20: Pull complete
a36abae8f559: Pull complete
aaad39e4e83: Pull complete
15c4963bdf8e: Pull complete
2b2f9df68344: Pull complete
0be2444c0a64: Pull complete
bba1a160bb54: Pull complete
695ab411b701: Pull complete
098e6cd28a3d: Pull complete
187d98c68c0a: Pull complete
210411f4e14: Pull complete
Digest: sha256:01ccf53ac68037a1dc5aff50bf2336e6a672532cbe2536dc91ccb864478
94ad
Status: Downloaded newer image for jenkins/jenkins:lts-jdk17
9a67ad1685b34f9d910df8d3c75c43d9819800fc9512903bfe4240ff521da3c5
```

3.2- Comprobar que el contenedor anterior esta funcionando. Indica la línea de comandos para realizar esta operación.

docker ps, lista los contenedores que están actualmente en ejecución, mostrando información de los contenedores: ID, imagen, nombre, estado y puertos...

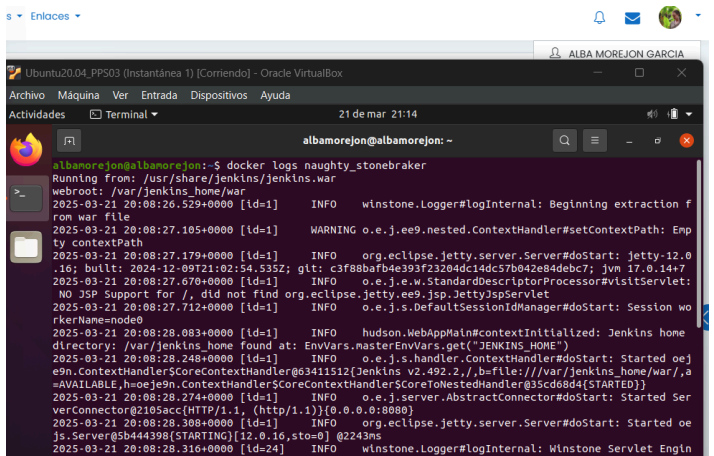


```
albamorejon@albamorejon:~$ docker ps
```

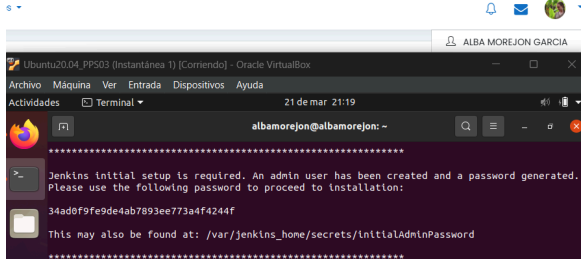
CONTAINER ID	IMAGE	COMMAND	CREATED	NAMES	STATUS	PORTS
7a277d1685b	jenkins/jenkins:lts-jdk17	"/usr/bin/tini -- /u..."	58 seconds ago	Up 51 seconds	0.0.0.0:8080->	
8080/tcp	naughty_stonebraker	"50000/tcp, ([:::50000]-50000/tcp	Up 2 minutes	127.0.0.1:4200		
ba7651ae7f3	ghcr.io/djaglins/dwa:latest	"docker-php-entrypol..."	6 weeks ago	Up 2 minutes		
->88/tcp	dwa_dbw_1					
0f04b20b205b	marlabs:10	"docker-entrypoint.s..."	6 weeks ago	Up 2 minutes	3306/tcp	
	dwa_db_1					

3.3- Mostrar el log del contenedor de Jenkins. Indica la línea de comandos para realizar esta operación.

docker logs naughty_stonebraker, obtenemos los logs de un contenedor Docker en ejecución llamado naughty_stonebraker.



```
albamorejon@albamorejon:~$ docker logs naughty_stonebraker
Running from: /usr/share/jenkins/jenkins.war
webroot: /var/jenkins_home/war
2025-03-21 20:08:26.529+0000 [id=1] INFO winstone.Logger#LogInternal: Beginning extraction f
ron war file
2025-03-21 20:08:27.105+0000 [id=1] WARNING o.e.jee9.nested.ContextHandler#setContextPath: Emp
ty contextPath
2025-03-21 20:08:27.179+0000 [id=1] INFO org.eclipse.jetty.server.Server#doStart: jetty-12.0
.16; built: 2024-12-09T21:02:54.535Z; git: c3f88baf4e393f23204dc14dc57b042e84dec7; jvm 17.0.14-7
2025-03-21 20:08:27.670+0000 [id=1] INFO o.e.j.e.w.StandardDescriptorProcessor#visitServlet:
NO JSP Support for /, did not find org.eclipse.jetty.ee9.jsp.JettyJspServlet
2025-03-21 20:08:27.712+0000 [id=1] INFO o.e.j.s.DefaultSessionIdManager#doStart: Session wo
rkerName=node0
2025-03-21 20:08:28.083+0000 [id=1] INFO hudson.WebAppMain#contextInitialized: Jenkins home
directory: /var/jenkins_home found at: EnvVars.masterEnvVars.get("JENKINS_HOME")
2025-03-21 20:08:28.248+0000 [id=1] INFO o.e.j.s.handler.ContextHandler#doStart: Started oej
ee9n.ContextHandler$CoreContextHandler@63411512{Jenkins v2.492.2,,bfile:///var/jenkins_home/war,/a
=AVAILABLE,h=oej9n.ContextHandler$CoreContextHandler$CoreNestedHandler@35cd68d4{STARTED}}
2025-03-21 20:08:28.274+0000 [id=1] INFO o.e.j.server.AbstractConnector#doStart: Started Ser
verConnector@2105acc(HTTP/1.1, (http/1.1)){0.0.0.0:8080}
2025-03-21 20:08:28.308+0000 [id=1] INFO org.eclipse.jetty.server.Server#doStart: Started oe
js.Server@b444398{STARTING}[12.0.16,sto=0] @2243ms
2025-03-21 20:08:28.316+0000 [id=24] INFO winstone.Logger#LogInternal: Winstone Servlet Engin
```



```
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

34ad0f9fe9de4ab7893ee773a4f4244f

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
```

Please use the following password to proceed to installation:
34ad0f9fe9de4ab7893ee773a4f4244f

3.4- Acceder a aplicación Jenkins desde un navegador

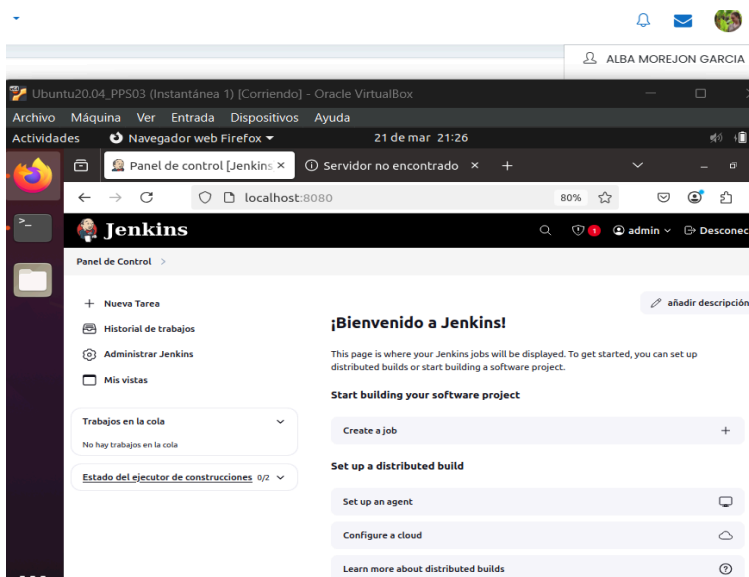
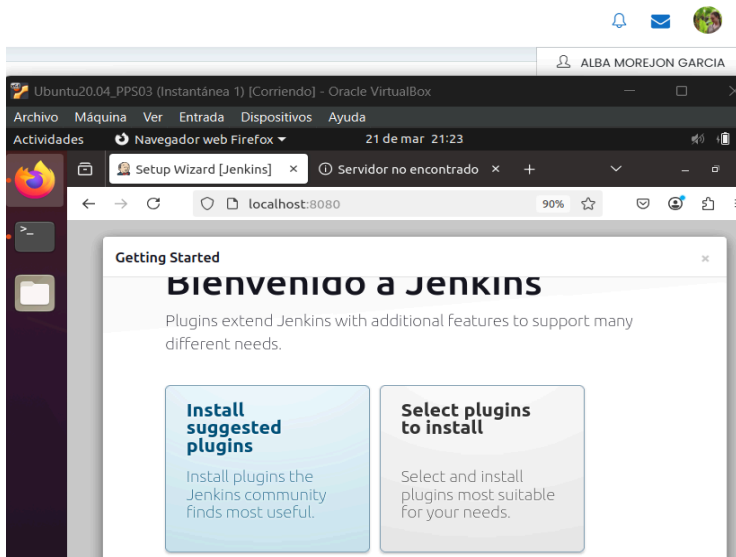
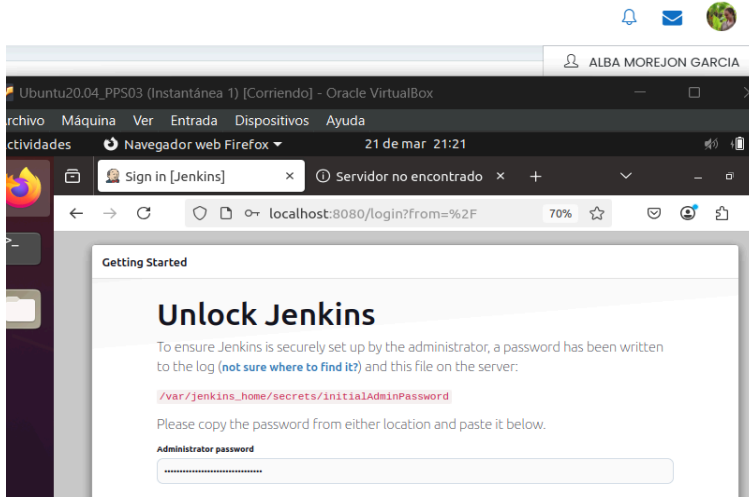
(<https://www.enkins.io/doc/book/installing/docker/>):

Abre un navegador con la dirección <http://localhost:8080> (si has puesto el 8080)

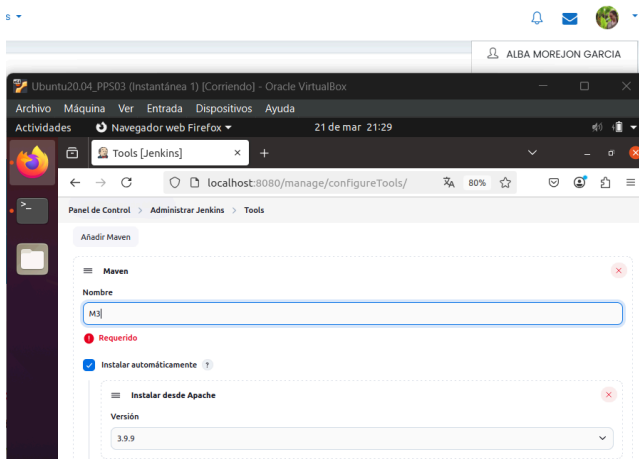
El token que pide aparece en el log del contenedor.

Escoge la opción de instalar los plugins recomendados.

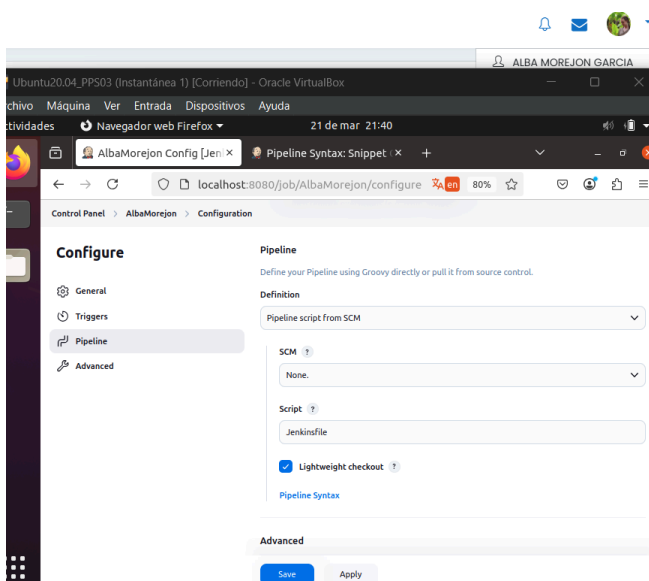
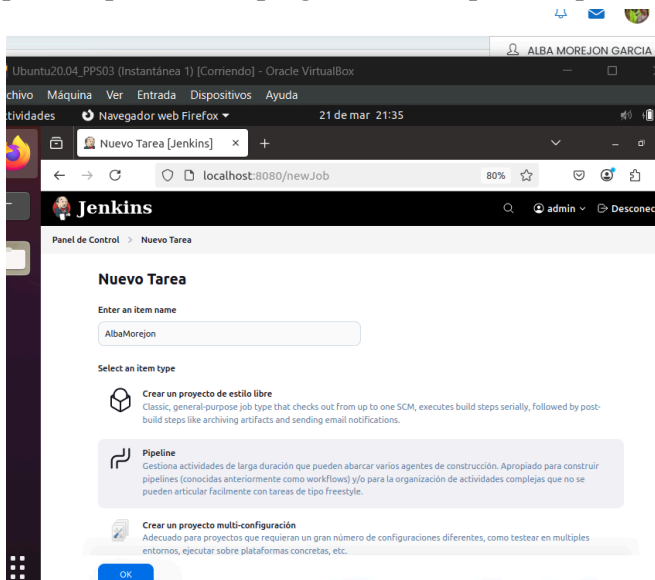
NO hace falta que crees un usuario (Skip and continue as admin)



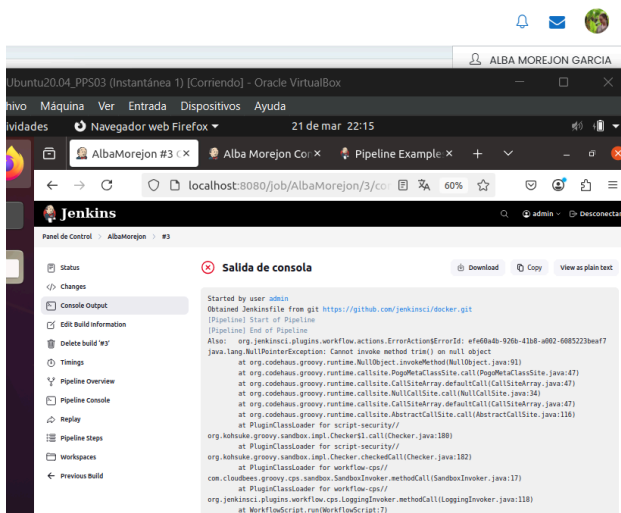
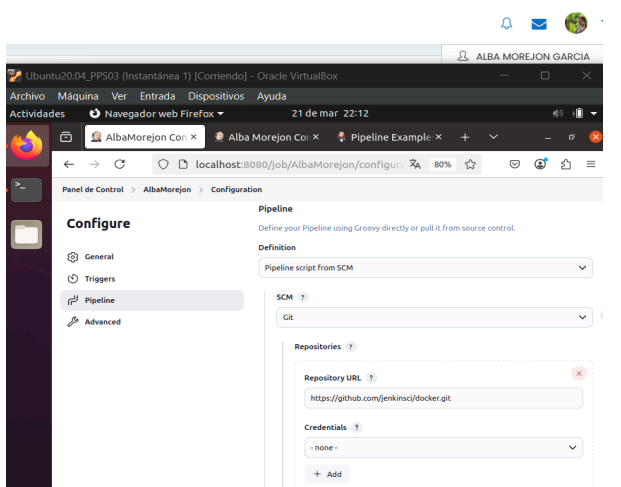
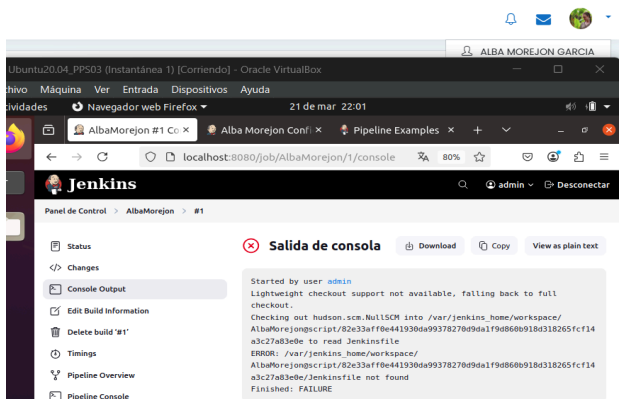
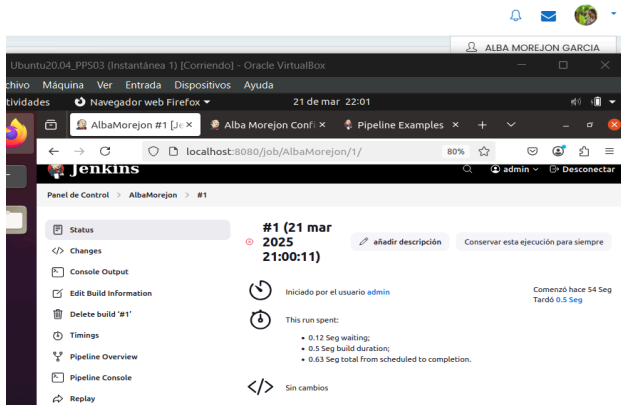
3.5- Instalar la tool de Maven con el nombre M3 dentro de Jenkins (Panel de control - Administrar Jenkins - Tools - Instalaciones de Maven - Añadir Maven - Nombre M3 y la opción de instalar automáticamente)



3.6- Crear un pipeline sencillo que se llame XXXXX (donde XXXXX es tu nombre): creas una tarea de tipo pipeline y en el apartado script seleccionas try sample Pipeline... escoges Github+ Maven. Copia el código del pipeline (a ese archivo se le denomina Jenkinsfile) en algún sitio porque te va a hacer falta para responder una pregunta en un apartado posterior

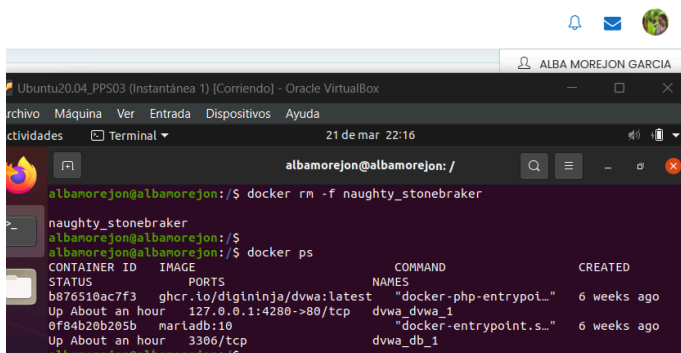


3.7- Ejecutar el pipeline anterior y mostrar el pantallazo de la salida.



3.8- Eliminar el contenedor de Jenkins. Indica la línea de comandos para realizar esta operación.

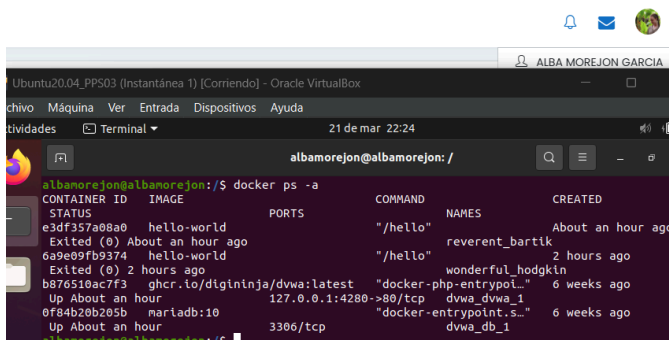
`docker rm -f naughty_stonebraker`, elimina contenedores docker (docker rm) forzando la eliminación si está en ejecución (-f) con el nombre de naughty_stonebraker



```
albamorejon@albamorejon:/$ docker rm -f naughty_stonebraker
naughty_stonebraker
albamorejon@albamorejon:/$
albamorejon@albamorejon:/$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED
STATUS        PORTS
b876510ac7f3   ghcr.io/digininja/dwa:latest       "docker-php-entrypoi..." 6 weeks ago
Up About an hour 127.0.0.1:4280->80/tcp   dwa_dwva_1
0f84b20b205b   mariadb:10                          "docker-entrypoint.s..." 6 weeks ago
Up About an hour 3306/tcp                dwa_db_1
```

3.9- Comprobar que el contenedor anterior ya no está funcionando. Indica la línea de comandos para realizar esta operación.

`docker ps -a`, muestra todos los contenedores, en cualquier estado y vemos que ya no aparece el jenkins



```
albamorejon@albamorejon:/$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED
STATUS        PORTS
e3df357a88a0   hello-world                          "/hello"                About an hour ago
Exited (0)
6a9e09fb9374   hello-world                          "/hello"                2 hours ago
Exited (0)
b876510ac7f3   ghcr.io/digininja/dwa:latest       "docker-php-entrypoi..." 6 weeks ago
Up About an hour 127.0.0.1:4280->80/tcp   dwa_dwva_1
0f84b20b205b   mariadb:10                          "docker-entrypoint.s..." 6 weeks ago
Up About an hour 3306/tcp                dwa_db_1
```

4- Responde a las siguientes preguntas

4.1- ¿Qué ventajas aporta trabajar con contenedores?

Ofrece ventajas como:

1. Portabilidad: Los contenedores son portables entre diferentes entornos, esto significa que puedes ejecutar el mismo contenedor en tu máquina local, en un servidor de pruebas o en la nube.
2. Aislamiento: Cada contenedor se ejecuta de forma aislada, lo que significa que los problemas en un contenedor no afectan a otros contenedores. Esto mejora la seguridad y la estabilidad.
3. Escalabilidad: Los contenedores permiten escalar aplicaciones fácilmente. Puedes aumentar o disminuir el número de contenedores según la demanda, lo que facilita la gestión de recursos.
4. Eficiencia: Los contenedores son más ligeros que las máquinas virtuales porque comparten el mismo sistema operativo. Esto reduce el uso de recursos y mejora el rendimiento.

4.2- Revisa el pipeline que has creado en el apartado anterior ¿Qué hace?

Buscando información encontramos este esquema de script:

Se clona el repositorio de GitHub especificado. Esto asegura que el código fuente está disponible para las siguientes etapas.

Se ejecuta el comando `mvn clean install` para compilar el proyecto usando Maven. Esto construye el proyecto y verifica que no haya errores de compilación.

Se ejecuta las pruebas del proyecto con `mvn test`. Esto asegura que el código funciona correctamente y que no hay errores en las pruebas unitarias.

Se ejecuta el comando `mvn deploy` para desplegar el proyecto. Esto puede implicar la publicación del artefacto en un repositorio de artefactos o la implementación en un servidor.

```
pipeline {
    agent any
    stages {
        stage('Checkout') {
            steps {
                git 'https://github.com/tu-repositorio/tu-proyecto.git'
            }
        }
        stage('Build') {
            steps {
                sh 'mvn clean install'
            }
        }
        stage('Test') {
            steps {
                sh 'mvn test'
            }
        }
        stage('Deploy') {
            steps {
                sh 'mvn deploy'
            }
        }
    }
}
```


4.3- Busca información sobre pruebas de seguridad en un pipeline e indica que pasos se podrían añadir al pipeline anterior para asegurar que la aplicación que se va a poner en producción es segura.

Para asegurar que la aplicación que se va a poner en producción es segura, puedes añadir:

- Análisis de código estático (SAST): integrar herramientas de análisis de código estático para detectar vulnerabilidades en el código fuente durante el desarrollo. Ejemplos de herramientas: SonarQube, Checkmarx 5.
- Análisis de dependencias: escanear las dependencias del proyecto para detectar vulnerabilidades conocidas. Ejemplos de herramientas: OWASP Dependency-Check, Snyk 6.
- Pruebas de seguridad dinámicas (DAST): realizar pruebas de seguridad dinámicas para detectar vulnerabilidades en la aplicación en ejecución. Ejemplos de herramientas: OWASP ZAP, Burp Suite 5.

Apartado 2: Dockerfile

1- Dado el siguiente archivo Dockerfile

```
FROM ubuntu
LABEL maintainer="admin@ejemplo.com"
ARG APP_DATO1=1
RUN apt-get update && apt-get install -y apache2
RUN mkdir /var/www/html/ejemplo
EXPOSE 80
ADD index2.html /var/www/html/
ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

Rellena la siguiente tabla donde tendrás que indicar para que sirve cada línea

FROM ubuntu	Especifica la imagen base que se utilizará para crear el contenedor. En este caso, se utiliza la imagen de Ubuntu.
LABEL maintainer="admin@ejemplo.com"	Añade una etiqueta con información del mantenedor de la imagen. Esto es útil para identificar quién es responsable de la imagen.
ARG APP_DATO1=1	Define una variable de construcción llamada APP_DATO1 con un valor predeterminado de 1. Las variables de construcción se pueden utilizar durante el proceso de construcción de la imagen.
RUN apt-get update && apt-get install -y apache2	Ejecuta comandos en el contenedor para actualizar la lista de paquetes e instalar el servidor web Apache.
RUN mkdir /var/www/html/ejemplo	Crea un directorio llamado ejemplo en la ruta /var/www/html/.
EXPOSE 80	Indica que el contenedor escuchará en el puerto 80 en tiempo de ejecución. Esto es útil para la documentación y para que las herramientas de orquestación sepan qué puertos exponer.
ADD index2.html /var/www/html/	Copia el archivo index2.html desde el contexto de construcción del Dockerfile al directorio /var/www/html/ dentro del contenedor.
ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]	Define el comando que se ejecutará cuando se inicie un contenedor a partir de esta imagen. En este caso, se inicia el servidor Apache en primer plano.

2- Responde a las siguientes preguntas y realiza las pruebas

2.1- ¿Para qué es útil un archivo Dockerfile?

Un archivo Dockerfile es útil para automatizar la creación de imágenes de Docker. Permite definir todos los pasos necesarios para construir una imagen, incluyendo la instalación de software, la configuración del entorno y la copia de archivos. Esto asegura que la imagen sea reproducible y consistente cada vez que se construya.

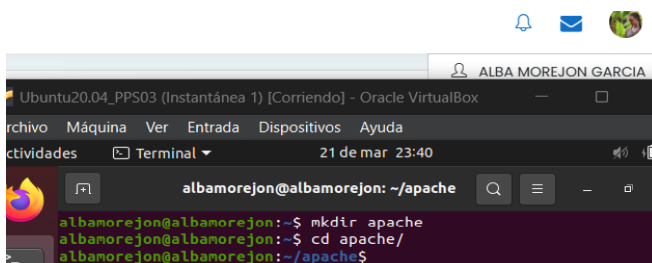
Un archivo Dockerfile es útil para definir la configuración y los pasos necesarios para crear una imagen de Docker. Es un script que contiene instrucciones sobre cómo construir una imagen, incluyendo la base de la imagen, los paquetes que se deben instalar, las configuraciones del sistema, y los comandos que se deben ejecutar.

2.2- ¿Con qué comando se crean las imágenes partiendo de un Dockerfile?

`docker build -t nombre_imagen:version .`, crea una imagen partiendo de un Dockerfile, asignando un nombre y una versión a la imagen (`-t imagen:version`) y se creará en el directorio actual (`.`).

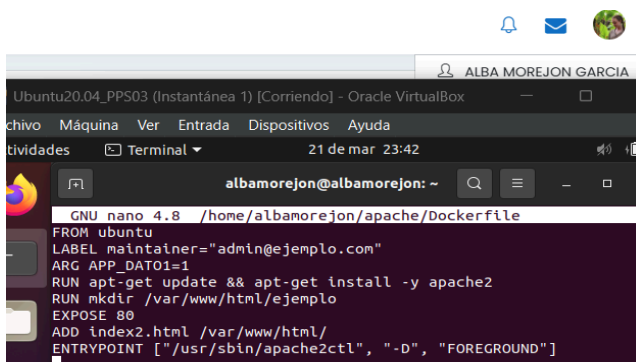
2.3- Realiza una prueba con el archivo anterior:

2.3.1- Crea un directorio



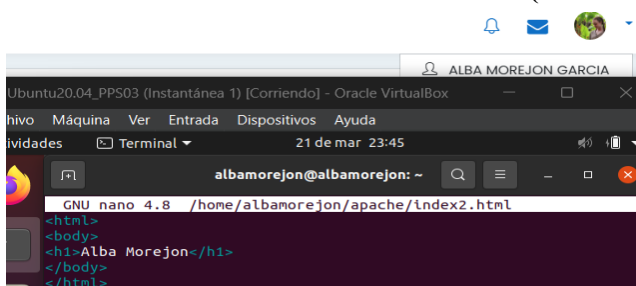
```
albamorejon@albamorejon: ~/apache
albamorejon@albamorejon:~$ mkdir apache
albamorejon@albamorejon:~$ cd apache/
albamorejon@albamorejon:~/apache$
```

2.3.2- Crea dentro del directorio el archivo Dockerfile anterior



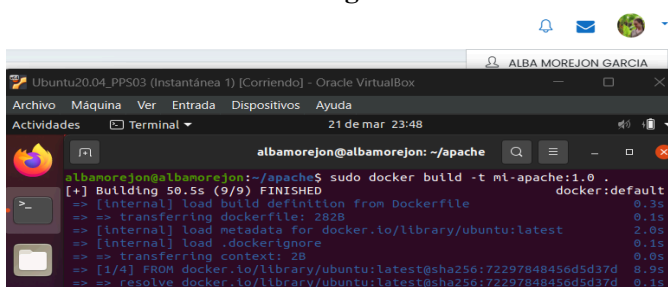
```
GNU nano 4.8 /home/albamorejon/apache/Dockerfile
FROM ubuntu
LABEL maintainer="admin@ejemplo.com"
ARG APP_DAT01=1
RUN apt-get update && apt-get install -y apache2
RUN mkdir /var/www/html/ejemplo
EXPOSE 80
ADD index2.html /var/www/html/
ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

2.3.3- Crea el archivo index2.html (escribe dentro tu nombre)



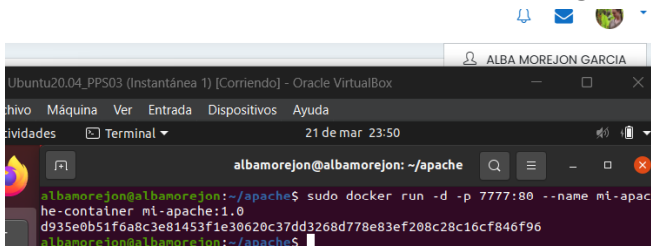
```
GNU nano 4.8 /home/albamorejon/apache/index2.html
<html>
<body>
<h1>Alba Morejon</h1>
</body>
</html>
```

2.3.4- Crea una imagen nueva con ese archivo Dockerfile de nombre mi-apache y versión 1.0

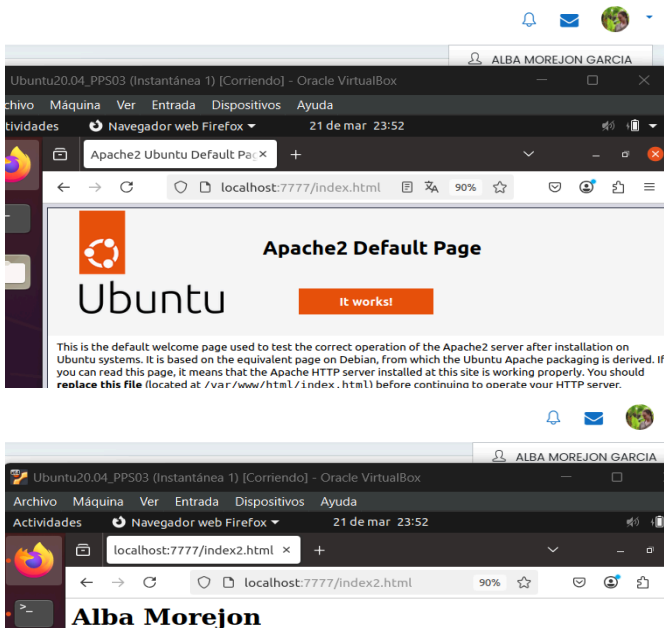


```
albamorejon@albamorejon: ~/apache
albamorejon@albamorejon:~/apache$ sudo docker build -t mi-apache:1.0 .
[*] Building 50.5s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [internal] load .dockerignore
=> [internal] load context: 28
=> [1/4] FROM docker.io/library/ubuntu:latest@sha256:72297848456d5d37d
=> resolve docker.io/library/ubuntu:latest@sha256:72297848456d5d37d
```

2.3.5- Lanza un contenedor con la imagen anterior en el puerto 7777.



2.3.6- Abre un navegador y accede al contenedor anterior mostrando las páginas index.html e index2.html



Apartado 3: Docker-compose

1- Rellena la siguiente tabla indicando para que sirve cada línea de comandos u opción dentro de un archivo docker-compose.yml

docker-compose up	Levanta y ejecuta todos los servicios definidos en el archivo docker-compose.yml. Si las imágenes no existen, las construye.
docker-compose down	Detiene y elimina los contenedores, redes y volúmenes definidos en el archivo docker-compose.yml.
services	Define los servicios que se ejecutarán en los contenedores. Cada servicio puede tener su propia configuración.
image	Especifica la imagen de Docker que se utilizará para el servicio. Puede ser una imagen existente en Docker Hub o una imagen personalizada.
build	Define la ruta del Dockerfile y el contexto para construir la imagen del servicio.
container-name	Asigna un nombre en específico al contenedor
ports	Mapea los puertos del contenedor a los puertos del host. Por ejemplo, - "8080: 80" mapea el puerto 80 del contenedor al puerto 8080 del host.

volumes	Monta volúmenes para persistencia de datos o para compartir archivos entre el host y el contenedor.
environment	Define variables de entorno que se pasarán al contenedor

2- Realiza las siguientes pruebas (incluye pantallazos de cada una)

2.1- Lanza dos contenedores con las siguientes líneas en una terminal y muestra pantallazos de las ejecución de esos comandos (docker ps -a)

```
docker run -d -p 33060:3306 --name mysql-db1 -e MYSQL_ROOT_PASSWORD=secret -v
```

```
mysql:/var/lib/mysql mysql:5.7.28
```

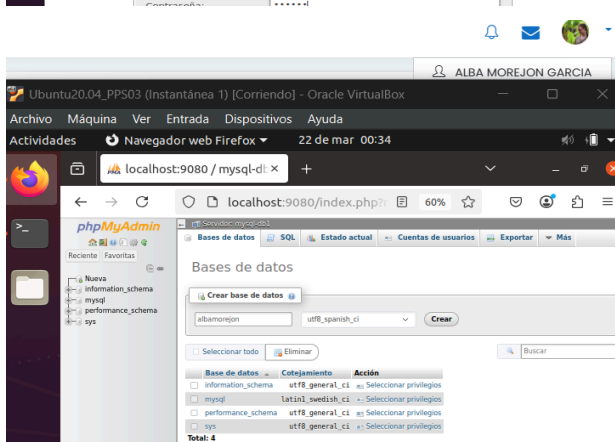
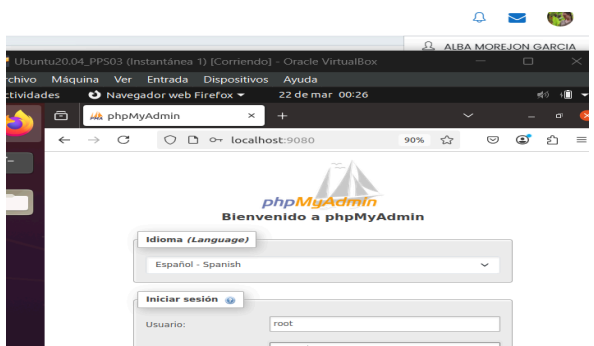
```
docker run -d --rm --name phpmyadminc --link mysql-db1 -e PMA_HOST=mysql-db1 -p 9080:80
```

```
phpmyadmin/phpmyadmin
```

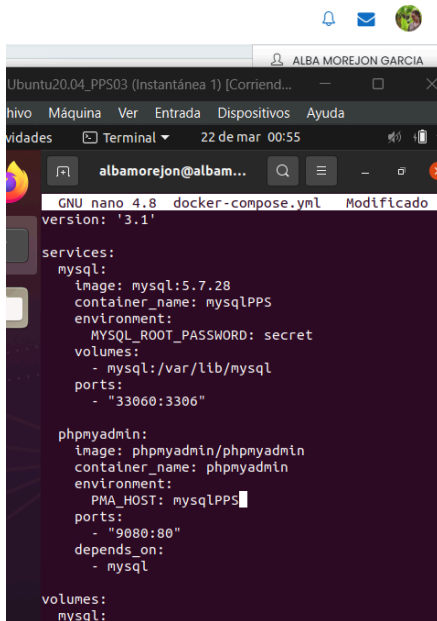
2.2- Utilizando el contenedor de phpmyadmin crea una base de datos con tu nombre en mysql.

Muestra pantallazos del navegador con el acceso a phpmyadmin y con la creación de la base de datos.

root:secret



2.3- Crea un archivo docker-compose.yml que permita lanzar esos dos contenedores utilizando docker compose. Incluye el contenido del archivo docker-compose.yml



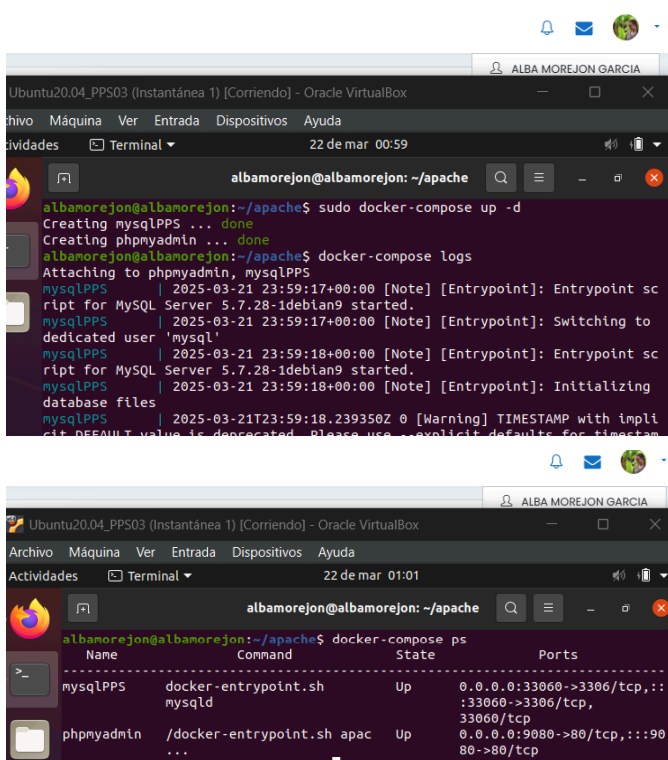
```
GNU nano 4.8 docker-compose.yml Modificado
version: '3.1'

services:
  mysql:
    image: mysql:5.7.28
    container_name: mysqlPPS
    environment:
      MYSQL_ROOT_PASSWORD: secret
    volumes:
      - mysql:/var/lib/mysql
    ports:
      - "33060:3306"

  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    container_name: phpmyadmin
    environment:
      PMA_HOST: mysqlPPS
    ports:
      - "9080:80"
    depends_on:
      - mysql

volumes:
  mysql:
```

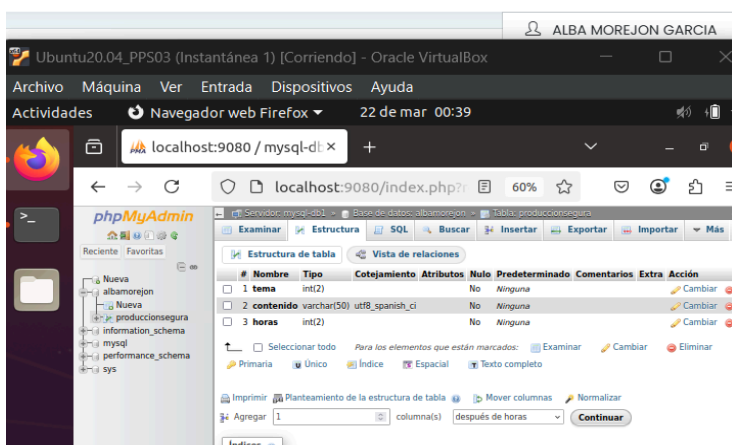
2.4- Lanza el archivo anterior. Muestra pantallazos de la ejecución en la consola



```
albamorejon@albamorejon: ~/apache
albamorejon@albamorejon:~/apache$ sudo docker-compose up -d
Creating mysqlPPS ... done
Creating phpmyadmin ... done
albamorejon@albamorejon:~/apache$ docker-compose logs
Attaching to phpmyadmin, mysqlPPS
mysqlPPS | 2025-03-21 23:59:17+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 5.7.28-1debian9 started.
mysqlPPS | 2025-03-21 23:59:17+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
mysqlPPS | 2025-03-21 23:59:18+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 5.7.28-1debian9 started.
mysqlPPS | 2025-03-21 23:59:18+00:00 [Note] [Entrypoint]: Initializing database files
mysqlPPS | 2025-03-21T23:59:18.239350Z 0 [Warning] TIMESTAMP with implicit DEFAULT value is deprecated. Please use --explicit-defaults-for-timestamp instead.

albamorejon@albamorejon: ~/apache
albamorejon@albamorejon:~/apache$ docker-compose ps
Name                Command             State              Ports
-----
mysqlPPS            docker-entrypoint.sh mysql               Up                  0.0.0.0:33060->3306/tcp,:::33060->3306/tcp
phpmyadmin          /docker-entrypoint.sh apac  Up                  0.0.0.0:9080->80/tcp,:::9080->80/tcp
```

2.5- Utilizando el contenedor de phpmyadmin crea una tabla en la base de datos. Muestra pantallazos del navegador con el acceso a phpmyadmin y con la creación de la tabla



2.6- Termina los contenedores lanzados. Muestra pantallazos de la ejecución en la consola

