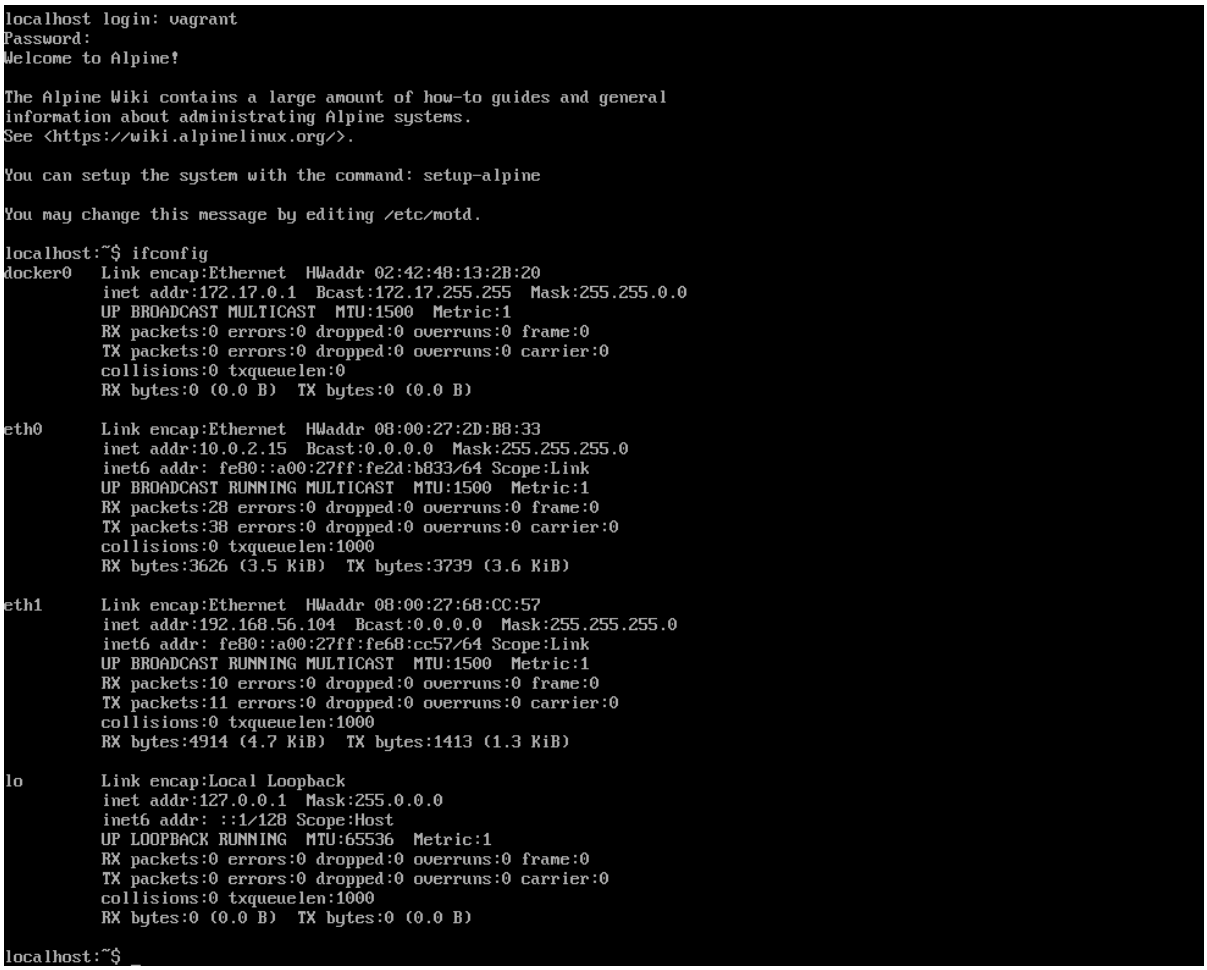
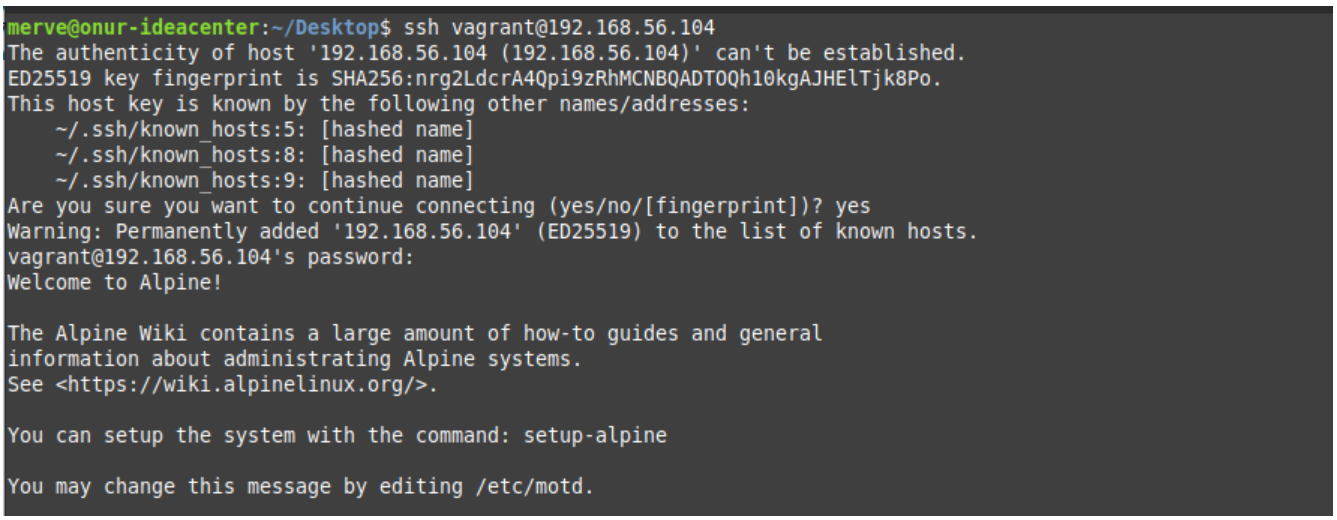
EJERCICIO 1

1. Creamos una máquina virtual con docker.



Conectamos con terminal.



2. Creamos y prueba una base de datos mariadb
a. Descargamos la imagen de mariadb con docker pull

```
localhost:~$ docker pull mariadb
Using default tag: latest
latest: Pulling from library/mariadb
a8b1c5f80c2d: Pull complete
b13b8cff7564: Pull complete
e5739d28aeee: Pull complete
0b3f8ae1fce9: Pull complete
61d4eb1159ff: Pull complete
a0b237c7c6ae: Pull complete
a6321fa47c19: Pull complete
12077f74b1db: Pull complete
Digest: sha256:3c43c6f4bc2931825c1207328429f422f30ce2cb029567da1df9a8cdcf2b8552
Status: Downloaded newer image for mariadb:latest
docker.io/library/mariadb:latest
localhost:~$
```

b. Creamos el contenedor de base de datos

```
localhost:~$ docker run --detach --name some-mariadb --env MARIADB_USER=wordpress --env MARIADB_PASSWORD=wordpress --env MARIADB_DATABASE=wordpress --env MARIADB_ROOT_PASSWORD=passwo
rd mariadb:latest
9c6bd70c0101a183ff3e5b91a3a57b8105201a3f198db5dac84be2bc69b8ede0
localhost:~$
```

Ejecutamos codigos:

docker run crea un contenedor
--name some-mariadb asigna un nombre para contenedor (nombre de contenedor = some-mariadb)

Variables de entorno:
-e (--env) MARIADB_ROOT_PASSWORD=password: Ésta es la contraseña del usuario root de la base de datos MARIADB.

Esto establece una variable de entorno llamada MARIADB_ROOT_PASSWORD dentro del contenedor con el valor "password".

-e (--env) MARIADB_DATABASE=wordpress: Esto crea una base de datos llamada wordpress dentro del contenedor MARIADB.

Esto establece una variable de entorno llamada MARIADB_DATABASE con el valor "wordpress".

-e (--env) MARIADB_USER=wordpress:

Esto establece una variable de entorno llamada MARIADB_USER con el valor "wordpress". Esto crea un usuario llamado wordpress para acceder a la base de datos.

-e (--env) MARIADB_PASSWORD=wordpress:

Esto establece una variable de entorno llamada MARIADB_PASSWORD con el valor "wordpress". Ésta es la contraseña del usuario wordpress para acceder a la base de datos.

-d (--detach): Ejecuta el contenedor en modo detached (separado), lo que significa que se ejecuta en segundo plano y no bloquea el terminal.

El comando **mariadb:latest** utiliza la imagen mariadb:latest de Docker Hub, la cual corresponde a la versión más reciente de la imagen oficial de MariaDB.

c. Creamos otro contenedor mariadb, conecta con el cliente mysql en línea de comandos y comprobamos que todo funciona correctamente, ejecutando alguna consulta

```
localhost:~$ docker network create wordpress
0d3eed9b7bad59f41c04cda8caab5c17f4370dc565f864f8e50cb983f3762aa9
localhost:~$
```

docker network: Este comando se utiliza para administrar redes de Docker.
create: Este comando crea específicamente una red nueva.
wordpress: Este es el nombre que asignamos a la red recién creada.

```
localhost:~$ docker run --detach --name my-mariadb --env MARIADB_USER=wordpress --env MARIADB_PASSWORD=wordpress --env MARIADB_DA
TABASE=wordpress --env MARIADB_ROOT_PASSWORD=password mariadb:latest
068f7f31e880a29cf976f8e8cb5c78029f4ca86ae51fa5c032dc2e5562f751fe
localhost:~$
```

Aquí creamos otro contenedor con el nombre **my-mariadb** y conactaremos a otro contenedor.

```
localhost:~$ docker run --name my-wordpress --network wordpress --link my-mariadb:db -d wordpress
Unable to find image 'wordpress:latest' locally
latest: Pulling from library/wordpress
Digest: sha256:f468bab53528df6f87dfe11a80de26eff57e0f515e243d9dec73a02c80c273a7
Status: Downloaded newer image for wordpress:latest
43e13b8fcbb6c66b830a567d0c7ae0007e0110b04f0526adee262315c16bd1cf
localhost:~$
```

Aquí este comando usa docker run para iniciar contenedores de WordPress.

--network wordpress: Conecta el contenedor a la red "wordpress".

--name my-wordpress: Asigna el nombre "my-wordpress" al contenedor.

--link my-mariadb:mydb: Vincula este contenedor a otro contenedor llamado "my-mariadb" con el alias "mydb". Esto permite que el contenedor de WordPress acceda al contenedor de la base de datos usando el alias mydb dentro de su configuración.

Este comando (docker exec -it my-mariadb mariadb -u root -p) nos permite conectarnos al servidor MariaDB que se ejecuta dentro del contenedor llamado "my-mariadb" con privilegios de root.

Mediante la consulta SHOW DATABASES; podemos verificar que la base de datos ‘wordpress’ existe.

```
localhost:~$ docker exec -it my-mariadb mariadb -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 6
Server version: 11.3.2-MariaDB-1:11.3.2+maria~ubu2204 mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| wordpress          |
+--------------------+
5 rows in set (0.000 sec)

MariaDB [(none)]>
```

3. Añadimos persistencia asociando el directorio con la base de datos a un directorio del host

```
localhost:~$ docker run --volume=/home/merve/Desktop/Infraestructura_para_big_data/Practica_5:/var/lib/mysql --env MARIADB_ROOT_PASSWORD=pass
word mariadb
2024-05-26 16:57:50+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server 1:11.3.2+maria~ubu2204 started.
2024-05-26 16:57:50+00:00 [Warn] [Entrypoint]: /sys/fs/cgroup///memory.pressure not writable, functionality unavailable to MariaDB
2024-05-26 16:57:50+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2024-05-26 16:57:50+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server 1:11.3.2+maria~ubu2204 started.
2024-05-26 16:57:51+00:00 [Note] [Entrypoint]: Initializing database files
2024-05-26 16:57:51 0 [Warning] mariadbd: io_uring_queue_init() failed with errno 1
2024-05-26 16:57:51 0 [Warning] InnoDB: liburing disabled: falling back to innodb_use_native_aio=OFF


PLEASE REMEMBER TO SET A PASSWORD FOR THE MariaDB root USER !
To do so, start the server, then issue the following command:

'/usr/bin/mariadb-secure-installation'

which will also give you the option of removing the test
databases and anonymous user created by default.  This is
strongly recommended for production servers.

See the MariaDB Knowledgebase at https://mariadb.com/kb

Please report any problems at https://mariadb.org/jira

The latest information about MariaDB is available at https://mariadb.org/.

Consider joining MariaDB's strong and vibrant community:
https://mariadb.org/get-involved/

2024-05-26 16:57:51+00:00 [Note] [Entrypoint]: Database files initialized
```

```
2024-05-26 16:57:53+00:00 [Note] [Entrypoint]: Temporary server stopped

2024-05-26 16:57:53+00:00 [Note] [Entrypoint]: MariaDB init process done. Ready for start up.

2024-05-26 16:57:53 0 [Note] Starting MariaDB 11.3.2-MariaDB-1:11.3.2+maria~ubu2204 source revision 068a6819eb63bcb01fdfa037c9bf3bf63c33ee42
as process 1
2024-05-26 16:57:53 0 [Note] InnoDB: Compressed tables use zlib 1.2.11
2024-05-26 16:57:53 0 [Note] InnoDB: Number of transaction pools: 1
2024-05-26 16:57:53 0 [Note] InnoDB: Using crc32 + pclmulqdq instructions
2024-05-26 16:57:53 0 [Note] mariadbd: O_TMPFILE is not supported on /tmp (disabling future attempts)
2024-05-26 16:57:53 0 [Warning] mariadbd: io_uring_queue_init() failed with errno 1
2024-05-26 16:57:53 0 [Warning] InnoDB: liburing disabled: falling back to innodb_use_native_aio=OFF
2024-05-26 16:57:53 0 [Note] InnoDB: Initializing buffer pool, total size = 128.000MiB, chunk size = 2.000MiB
2024-05-26 16:57:53 0 [Note] InnoDB: Completed initialization of buffer pool
2024-05-26 16:57:53 0 [Note] InnoDB: File system buffers for log disabled (block size=512 bytes)
2024-05-26 16:57:53 0 [Note] InnoDB: End of log at LSN=47875
2024-05-26 16:57:53 0 [Note] InnoDB: Opened 3 undo tablespaces
2024-05-26 16:57:53 0 [Note] InnoDB: 128 rollback segments in 3 undo tablespaces are active.
2024-05-26 16:57:53 0 [Note] InnoDB: Setting file './ibtmp1' size to 12.000MiB. Physically writing the file full; Please wait ...
2024-05-26 16:57:53 0 [Note] InnoDB: File './ibtmp1' size is now 12.000MiB.
2024-05-26 16:57:53 0 [Note] InnoDB: log sequence number 47875; transaction id 16
2024-05-26 16:57:53 0 [Note] Plugin 'FEEDBACK' is disabled.
2024-05-26 16:57:53 0 [Note] Plugin 'wsrep-provider' is disabled.
2024-05-26 16:57:53 0 [Note] InnoDB: Loading buffer pool(s) from /var/lib/mysql/ib_buffer_pool
2024-05-26 16:57:53 0 [Note] InnoDB: Buffer pool(s) load completed at 240526 16:57:53
2024-05-26 16:57:53 0 [Note] Server socket created on IP: '0.0.0.0'.
2024-05-26 16:57:53 0 [Note] Server socket created on IP: '::'.
2024-05-26 16:57:53 0 [Note] mariadbd: Event Scheduler: Loaded 0 events
2024-05-26 16:57:53 0 [Note] mariadbd: ready for connections.
Version: '11.3.2-MariaDB-1:11.3.2+maria~ubu2204'  socket: '/run/mysqld/mysqld.sock'  port: 3306  mariadb.org binary distribution
```

Este comando inicia un contenedor de MariaDB y establece la contraseña del usuario root.
Luego monta un volumen para persistir los datos. El volumen mapea el directorio de la máquina host (/home/merve/Desktop/Infraestructura_para_big_data/Practica_5) a un directorio dentro del contenedor (/var/lib/mysql).

```
localhost:~$ docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED              STATUS              PORTS      NAMES
5d978251c1e9   mariadb            "docker-entrypoint.s…"   About a minute ago   Up About a minute   3306/tcp   great_goldstine
43e13b8fcbb6   wordpress          "docker-entrypoint.s…"   19 minutes ago       Up 19 minutes       80/tcp     my-wordpress
068f7f31e880   mariadb:latest     "docker-entrypoint.s…"   About an hour ago    Up About an hour    3306/tcp   my-mariadb
9c6bd70c0101   mariadb:latest     "docker-entrypoint.s…"   About an hour ago    Up About an hour    3306/tcp   some-mariadb
```

docker ps enumera todos los contenedores que se están ejecutando actualmente en el host de Docker.

EJERCICIO 2

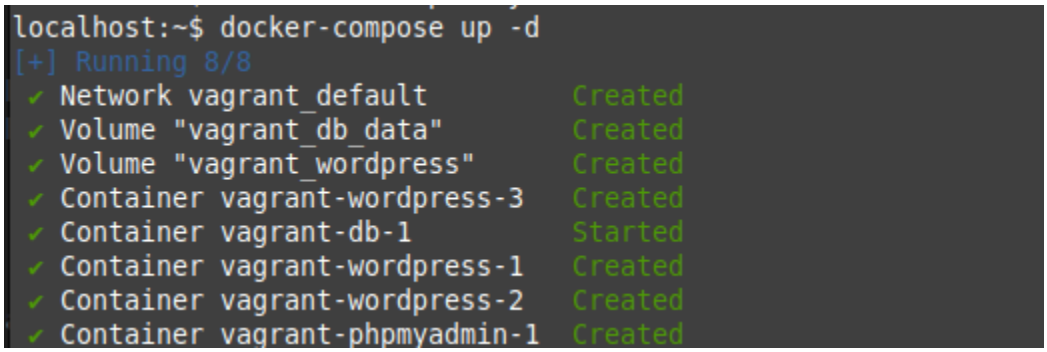
Crea la definición de docker-compose para wordpress, junto con una base de datos, phpmyadmin, y un volumen de datos. Se recomienda ir añadiendo elementos de forma incremental

1. Creamos la base de datos, mysql o mariadb, y un contenedor para los datos de la base de datos

2. Añadimos la herramienta de administración phpmyadmin

3. Añadimos wordpress

```
localhost:~$ cat docker-compose.yml
version: '3'

services:
  db:
    image: mariadb:latest
    environment:
      MYSQL_ROOT_PASSWORD: my-secret-pw
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress
    volumes:
      - db_data:/var/lib/mysql

  phpmyadmin:
    image: phpmyadmin
    ports:
      - 8080:80
    environment:
      PMA_ARBITRARY: 1
    depends_on:
      - db

  wordpress:
    image: wordpress
    ports:
      - 8081:80
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
    volumes:
      - wordpress:/var/www/html
    deploy:
      replicas: 3

volumes:
  wordpress:
  db_data:
```



En el archivo docker-compose.yml definimos una configuración. Creamos una base de datos con MariaDB y añadimos un contenedor para la herramienta de administración phpMyAdmin y otro para WordPress. Ejecutamos el comando docker-compose up -d para iniciar los contenedores y observamos que se han creado y comenzado a ejecutarse.

EJERCICIO 3

1. Creamos un cluster siguiendo los pasos indicados anteriormente, creando 3 nodos.

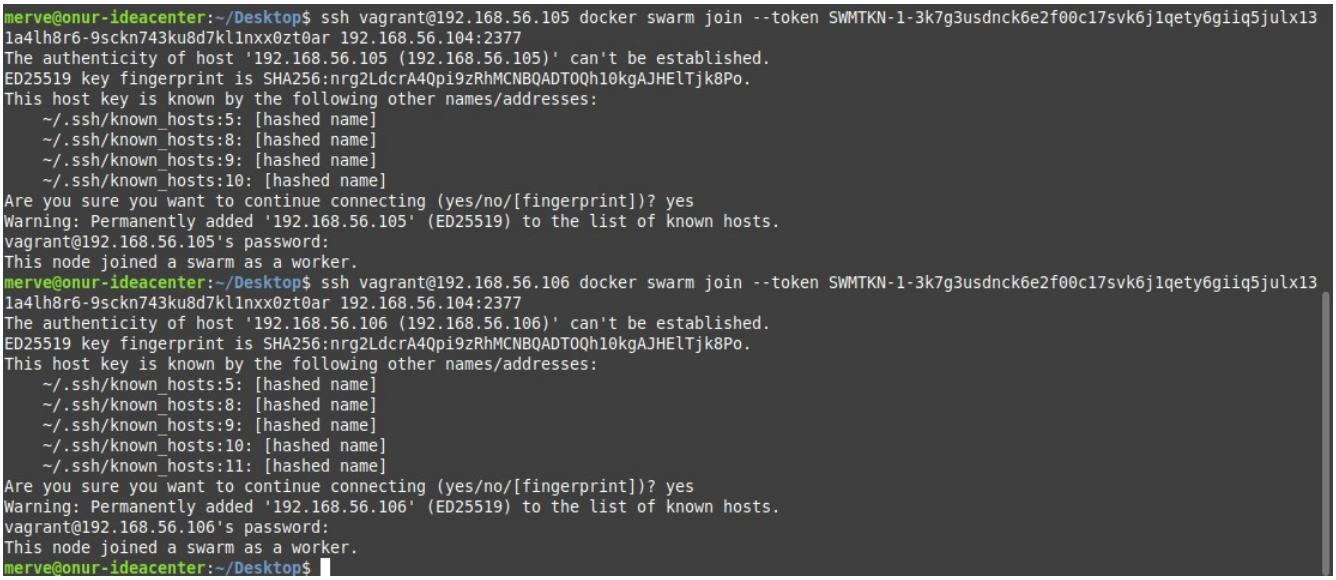

```
merve@onur-ideacenter:~/Desktop$ ssh vagrant@192.168.56.104
The authenticity of host '192.168.56.104 (192.168.56.104)' can't be established.
ED25519 key fingerprint is SHA256:nrg2LdcrA4Qpi9zRhMCNBQADTOQh10kgAJHElTjk8Po.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:5: [hashed name]
    ~/.ssh/known_hosts:8: [hashed name]
    ~/.ssh/known_hosts:9: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.104' (ED25519) to the list of known hosts.
vagrant@192.168.56.104's password:
Welcome to Alpine!

The Alpine Wiki contains a large amount of how-to guides and general
information about administrating Alpine systems.
See <https://wiki.alpinelinux.org/>.

You can setup the system with the command: setup-alpine

You may change this message by editing /etc/motd.
```

```
localhost:~$ docker swarm init --advertise-addr 192.168.56.104
Swarm initialized: current node (fj40t9rky8hk7ivsi8u0a1m13) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3k7g3usdnck6e2f00c17svk6j1qety6giiq5julx131a4lh8r6-9sckn743ku8d7kl1nxx0zt0ar 192.168.56.104:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

localhost:~$
```

docker swarm init: Este comando inicia el proceso de creación de un nuevo cluster Swarm en la máquina actual.

--advertise-addr 192.168.56.104: Esta opción específica la dirección IP que se anunciará a otros demonios Docker como la dirección del nodo administrador del Swarm. Esto permite que otras máquinas se unan al cluster utilizando esta dirección IP.

La máquina se convierte en el nodo administrador del cluster Swarm.



docker swarm join: Este comando le indica al demonio Docker que se una a un cluster Swarm existente(192.168.56.104).

--token SWMTKN-1-3k7g3usdnck6e2f00c17svk6j1qety6giiq5julx131a4lh8r6-9sckn743ku8d7kl1nxx0zt0ar: Esta opción proporciona el token necesario para unirse al cluster Swarm. El token suministrado funciona como una clave segura para ser miembro del cluster.

192.168.56.104:2377: Aquí se especifica la dirección del nodo administrador del cluster Swarm. La dirección IP (192.168.56.104) identifica el nodo administrador, y el puerto 2377 es el puerto predeterminado utilizado para la comunicación de Swarm.

Las máquinas (con IP de 192.168.56.105, 192.168.56.106) se transformarán en nodos trabajadores dentro del clúster Swarm, bajo la administración del nodo gestor ubicado en 192.168.56.104. Este nodo trabajador estará listo para ejecutar tareas y servicios distribuidos por el gestor de Swarm.

```
merve@onur-ideacenter:~/Desktop$ ssh vagrant@192.168.56.104 docker node ls
vagrant@192.168.56.104's password:
ID                            HOSTNAME    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
fj40t9rky8hk7ivsi8u0a1m13 *   localhost   Ready     Active         Leader           25.0.3
r3zlus29upfbfdn63xnfur6a3     localhost   Ready     Active                          25.0.3
vxrb4ur89ifvl3s4gzmbasnrv     localhost   Ready     Active                          25.0.3
merve@onur-ideacenter:~/Desktop$
```

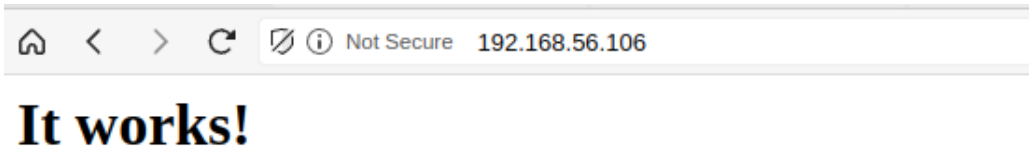
Podemos observar los nodos ‘Leader’ and ‘Worker’.

Inicialmente, probamos únicamente con la configuración web.

```
localhost:~$ cat docker-compose.yml
version: '3'
services:
  web:
    image: httpd:2.4-alpine
    ports:
      - "80:80"
    deploy:
      replicas: 3
localhost:~$ docker stack deploy -c docker-compose.yml web
Creating network web_default
Creating service web_web
```

```
localhost:~$ docker node ls
ID                            HOSTNAME    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
fj40t9rky8hk7ivsi8u0a1m13 *   localhost   Ready     Active         Leader           25.0.3
r3zlus29upfbfdn63xnfur6a3     localhost   Ready     Active                          25.0.3
vxrb4ur89ifvl3s4gzmbasnrv     localhost   Ready     Active                          25.0.3
localhost:~$ docker service ls
ID             NAME      MODE         REPLICAS   IMAGE              PORTS
ub6pm3noqlgy   web_web   replicated   3/3        httpd:2.4-alpine   *:80->80/tcp
localhost:~$
```

Ejecutamos el comando **docker stack deploy -c docker-compose.yml web**
Y observamos que todos los nodos funciona:



2. Añadimos la configuración de despliegue al ejercicio 2, de forma que wordpress esté replicado en los tres nodos.

Posteriormente, añadimos las configuraciones detalladas en el ejercicio 2.

```
localhost:~$ cat docker-compose.yml
version: '3'

services:
  db:
    image: mariadb:latest
    environment:
      MYSQL_ROOT_PASSWORD: my-secret-pw
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress
    volumes:
      - db_data:/var/lib/mysql

  phpmyadmin:
    image: phpmyadmin
    ports:
      - 8080:80
    environment:
      PMA_ARBITRARY: 1
    depends_on:
      - db

  wordpress:
    image: wordpress
    ports:
      - 8081:80
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
    volumes:
      - wordpress:/var/www/html
    deploy:
      replicas: 3

  web:
    image: httpd_2.4-alpine
    ports:
      - "80:80"
    deploy:
      replicas: 3

volumes:
  wordpress:
  db_data:
```

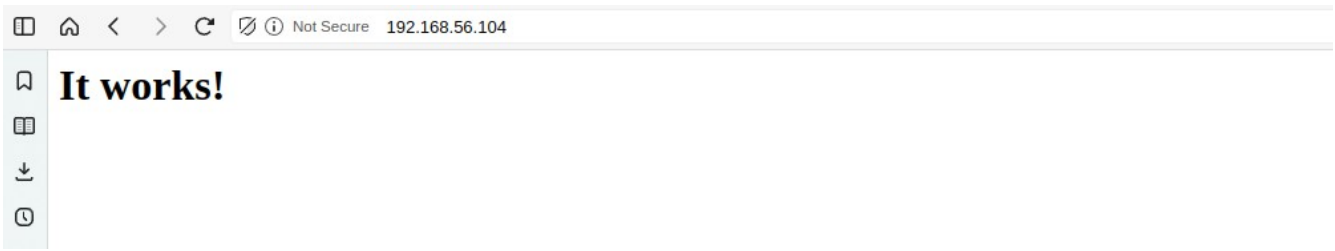
3. Comprobamos que funciona correctamente conectando a cualquier nodo del cluster.

```
localhost:~$ docker stack deploy -c docker-compose.yml web
Creating service web_phpmyadmin
Creating service web_wordpress
Updating service web_web (id: ub6pm3noqlgy8d6fxzm64hu2z)
image httpd_2.4-alpine:latest could not be accessed on a registry to record
its digest. Each node will access httpd_2.4-alpine:latest independently,
possibly leading to different nodes running different
versions of the image.

Creating service web_db
localhost:~$
```

```
localhost:~$ docker stack down web
Removing service web_db
Removing service web_phpmyadmin
Removing service web_web
Removing service web_wordpress
Removing network web_default
```

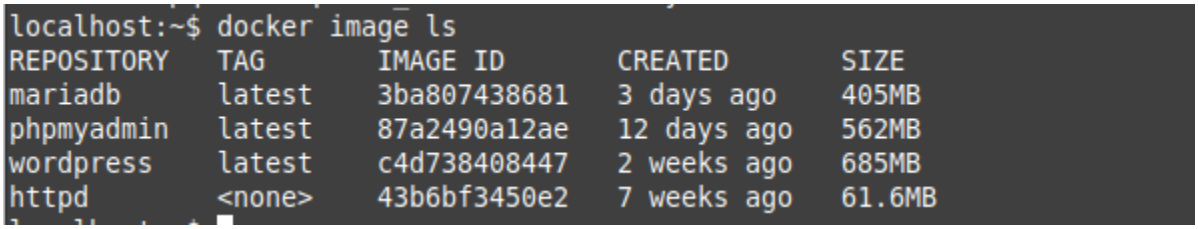
```
localhost:~$ docker stack deploy -c docker-compose.yml wordpress
Creating service wordpress_web
Updating service wordpress_db (id: ucice6igo8mykr27si320yibn)
Creating service wordpress_phpmyadmin
Creating service wordpress_wordpress
localhost:~$
```

En ambos casos, se verifica que el funcionamiento es correcto al conectarse a cualquier nodo del clúster.




```
localhost:~$ docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED       STATUS       PORTS      NAMES
37c5f183c9b0   mariadb:latest     "docker-entrypoint.s…"   2 hours ago   Up 2 hours   3306/tcp   vagrant-db-1
5d978251c1e9   mariadb            "docker-entrypoint.s…"   3 hours ago   Up 3 hours   3306/tcp   great_goldstine
43e13b8fcbb6   wordpress          "docker-entrypoint.s…"   3 hours ago   Up 3 hours   80/tcp     my-wordpress
068f7f31e880   mariadb:latest     "docker-entrypoint.s…"   4 hours ago   Up 4 hours   3306/tcp   my-mariadb
9c6bd70c0101   mariadb:latest     "docker-entrypoint.s…"   4 hours ago   Up 4 hours   3306/tcp   some-mariadb
8a28553f5d68   wordpress:latest   "docker-entrypoint.s…"   4 hours ago   Up 4 hours   80/tcp     wordpress_wordpress.3.mpetffaewswg9abkz12ymtjzm
519f6bf4c243   phpmyadmin:latest  "/docker-entrypoint.…"   4 hours ago   Up 4 hours   80/tcp     wordpress_phpmyadmin.1.k2fjemmwqwc5ndot749nfuuhl
localhost:~$ docker service ls
ID             NAME                   MODE         REPLICAS   IMAGE                    PORTS
ucice6igo8my   wordpress_db           replicated   1/1        mariadb:latest
eijfb8nyelvh   wordpress_phpmyadmin   replicated   1/1        phpmyadmin:latest        *:8080->80/tcp
kjcbl1l1gi2f   wordpress_web          replicated   0/3        httpd_2.4-alpine:latest  *:80->80/tcp
3s2fg2sgt5b8   wordpress_wordpress    replicated   3/3        wordpress:latest         *:8081->80/tcp
```

```
localhost:~$ docker network ls
NETWORK ID     NAME                DRIVER    SCOPE
d13975722a95   bridge              bridge    local
8d5f35014bf4   docker_gwbridge     bridge    local
91c7b1a7e2c1   host                host      local
qty6cw21db5h   ingress             overlay   swarm
4add795da367   none                null      local
3047d6158909   vagrant_default     bridge    local
0d3eed9b7bad   wordpress           bridge    local
9nrt2ax75npq   wordpress_default   overlay   swarm
```



Aquí podemos ver todos los servicios, contenedores, redes e imágenes que hemos creado. Observamos que todo funciona correctamente.