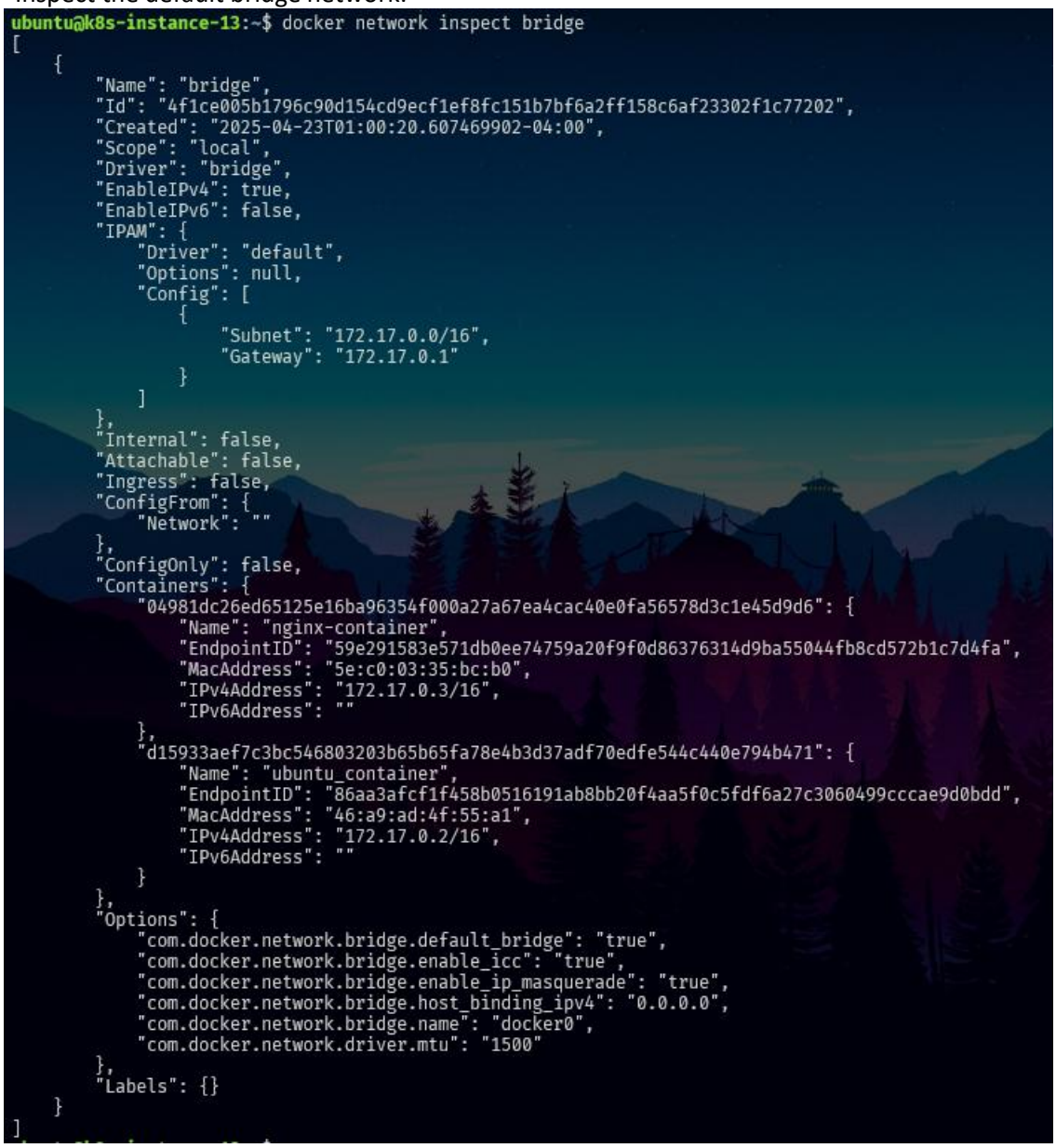
Homework – Docker Extended Topics

Exercise 1

1. List all Docker networks.

```
ubuntu@k8s-instance-13:~$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
4f1ce005b179   bridge    bridge    local
4ab530c8bc2a   host      host      local
8db1d9f6eeed   none      null      local
```

2. Inspect the default bridge network.



```
ubuntu@k8s-instance-13:~$ docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "4f1ce005b1796c90d154cd9ecf1ef8fc151b7bf6a2ff158c6af23302f1c77202",
        "Created": "2025-04-23T01:00:20.607469902-04:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv4": true,
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "04981dc26ed65125e16ba96354f000a27a67ea4cac40e0fa56578d3c1e45d9d6": {
                "Name": "nginx-container",
                "EndpointID": "59e291583e571db0ee74759a20f9f0d86376314d9ba55044fb8cd572b1c7d4fa",
                "MacAddress": "5e:c0:03:35:bc:b0",
                "IPv4Address": "172.17.0.3/16",
                "IPv6Address": ""
            },
            "d15933aef7c3bc546803203b65b65fa78e4b3d37adf70edfe544c440e794b471": {
                "Name": "ubuntu_container",
                "EndpointID": "86aa3afcf1f458b0516191ab8bb20f4aa5f0c5fdf6a27c3060499cccae9d0bdd",
                "MacAddress": "46:a9:ad:4f:55:a1",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
```

3. Create a new bridge user-defined network. (en la misma imagen que el 4)
4. Run a container attached to it and inspect its IP. (volvemos a revisar la red para ver la config)

```
ubuntu@k8s-instance-13:~$ docker network create --driver bridge my-bridge-network
5dc8906463a522c2b59daa430bd3ddd3508d116c5d13be5b7ec02e9e2581b00a
ubuntu@k8s-instance-13:~$ docker run --name nginx-net --network my-bridge-network -d nginx
96e56681724141755526f77e86c55e8a2191d6c333d47cfecfc55a078870c766f
ubuntu@k8s-instance-13:~$ docker network inspect my-bridge-network
[
    {
        "Name": "my-bridge-network",
        "Id": "5dc8906463a522c2b59daa430bd3ddd3508d116c5d13be5b7ec02e9e2581b00a",
        "Created": "2025-05-02T22:09:41.12007494-04:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv4": true,
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "96e56681724141755526f77e86c55e8a2191d6c333d47cfecfc55a078870c766f": {
                "Name": "nginx-net",
                "EndpointID": "5f6975209ee8a474cf10d7660ea35682f9bc8a71abf35ed11de0e199f67add27",
                "MacAddress": "b6:eb:e5:7d:f4:d8",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

Exercise 2

1. Run two Nginx containers which have to be connected to that user-defined network created in Exercise 1.

```
ubuntu@k8s-instance-13:~$ docker run -d --name nginx1 --network my-bridge-ne
twork nginx
5a1703c4b2425956a105c789bcef7f1fef947b7e17de68edc27005e2c2b6c289
ubuntu@k8s-instance-13:~$ docker run -d --name nginx2 --network my-bridge-ne
twork nginx
f39daee0a878f017b942826326ba662a04609791c16982d053a8f11ee6a62ec5
```

2. Use ping within both containers to test communication each other by container name

```
ubuntu@k8s-instance-13:~$ docker exec nginx1 ping -c 4 nginx2
OCI runtime exec failed: exec failed: unable to start container process: exe
c: "ping": executable file not found in $PATH: unknown
```

```
ubuntu@k8s-instance-13:~$ docker exec nginx1 ping -c 4 nginx2
PING nginx2 (172.18.0.4) 56(84) bytes of data.
64 bytes from nginx2.my-bridge-network (172.18.0.4): icmp_seq=1 ttl=64 time=
0.424 ms
64 bytes from nginx2.my-bridge-network (172.18.0.4): icmp_seq=2 ttl=64 time=
0.150 ms
64 bytes from nginx2.my-bridge-network (172.18.0.4): icmp_seq=3 ttl=64 time=
0.097 ms
64 bytes from nginx2.my-bridge-network (172.18.0.4): icmp_seq=4 ttl=64 time=
0.100 ms

── nginx2 ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3012ms
rtt min/avg/max/mdev = 0.097/0.192/0.424/0.135 ms
ubuntu@k8s-instance-13:~$ docker exec nginx2 ping -c 4 nginx1
PING nginx1 (172.18.0.3) 56(84) bytes of data.
64 bytes from nginx1.my-bridge-network (172.18.0.3): icmp_seq=1 ttl=64 time=
0.063 ms
64 bytes from nginx1.my-bridge-network (172.18.0.3): icmp_seq=2 ttl=64 time=
0.120 ms
64 bytes from nginx1.my-bridge-network (172.18.0.3): icmp_seq=3 ttl=64 time=
0.132 ms
64 bytes from nginx1.my-bridge-network (172.18.0.3): icmp_seq=4 ttl=64 time=
0.149 ms

── nginx1 ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 0.063/0.116/0.149/0.032 ms
```

Exercise 3

(Todo en una sola imagen)

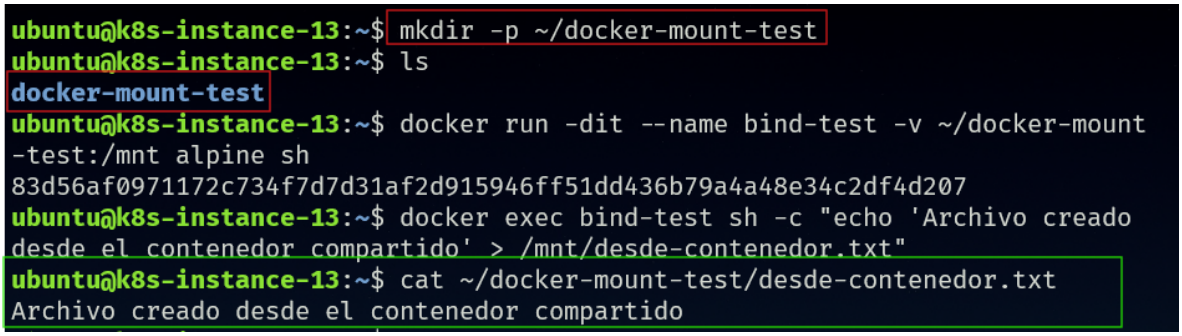
1. Create a Docker volume: mydata
2. Run a container using the volume.
3. Write a file inside /data from the container, then:
    a. Stop the container.
    b. Relaunch to verify persistence.

```
ubuntu@k8s-instance-13:~$ docker volume create mydata
mydata
ubuntu@k8s-instance-13:~$ docker run -dit --name test-volume --mount source=
mydata,target=/data alpine sh
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
f18232174bc9: Pull complete
Digest: sha256:a8560b36e8b8210634f77d9f7f9efd7ffa463e380b75e2e74aff4511df3ef
88c
Status: Downloaded newer image for alpine:latest
52d49bfe54f6eb4da94de3e8dba701a1e75a358dfe0bdc0574e1d634003413e7
ubuntu@k8s-instance-13:~$ docker exec test-volume sh -c "echo 'Hola desde el
 volumen de prueba my data' > /data/archivo.txt"
ubuntu@k8s-instance-13:~$ docker stop test-volume
test-volume
ubuntu@k8s-instance-13:~$ docker start test-volume
test-volume
ubuntu@k8s-instance-13:~$ docker exec test-volume cat /data/archivo.txt
Hola desde el volumen de prueba my data
ubuntu@k8s-instance-13:~$
```

Exercise 4

(Toda en una sola imagen)

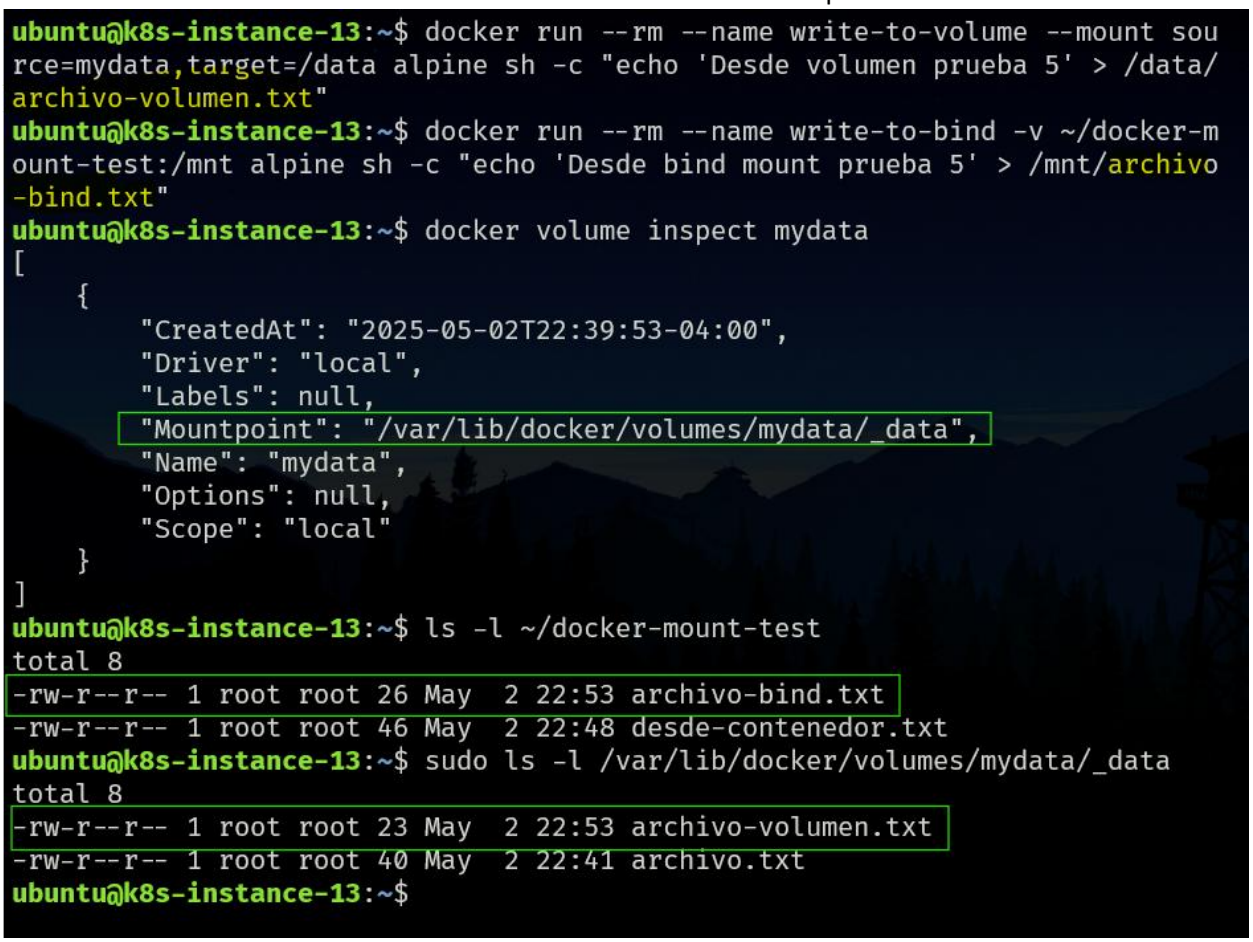
1. Create a directory on your host.
2. Run a container with a bind mount.
3. Create a file in /mnt from the container and check the host



Exercise 5

(Toda en una sola imagen)

1. Create a file in a named volume.
2. Create a file using a bind mount.
3. Observe where data is stored on the host with docker volume inspect and ls.



Exercise 6

1. Run an Ubuntu container with the necessary options to enable Docker in Docker (DinD).

```
ubuntu@k8s-instance-13:~$ docker run -dit --name dind-ubuntu --privileged -v
 /var/run/docker.sock:/var/run/docker.sock ubuntu bash
fe622e22337b23f672d15e5a2483584b5ec6689158bfc03d498dbc199462d65e
ubuntu@k8s-instance-13:~$ docker exec dind-ubuntu apt-get update
Get:1 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
```

```
ubuntu@k8s-instance-13:~$ docker exec dind-ubuntu apt-get install -y docker.
io
Reading package lists ...
Building dependency tree ...
Reading state information ...
The following additional packages will be installed:
  adduser apparmor bridge-utils ca-certificates containerd dbus dbus-bin
  dbus-daemon dbus-session-bus-common dbus-system-bus-common dns-root-data
  dnsmasq-base git git-man iproute2 iptables krb5-locales less libapparmor1
  libatm1t64 libbpf1 libbrotli1 libbsd0 libcap2-bin libcbor0.10
  libcurl3t64-gnutls libdbus-1-3 libedit2 libelf1t64 liberror-perl libexpat1
  libfido2-1 libgdbm-compat4t64 libgdbm6t64 libgssapi-krb5-2 libip4tc2
  libip6tc2 libjansson4 libk5crypto3 libkeyutils1 libkrb5-3 libkrb5support0
  libldap-common libldap2 libmnl0 libnetfilter-conntrack3 libnfnetlink0
  libnftables1 libnftnl11 libnghttp2-14 libpam-cap libperl5.38t64 libpsl5t64
  librtmp1 libsasl2-2 libsasl2-modules libsasl2-modules-db libssh-4
  libtirpc-common libtirpc3t64 libx11-6 libx11-data libxau6 libxcb1 libxdmcp
6
  libxext6 libxmuu1 libxtables12 netbase netcat-openbsd nftables
  openssh-client openssl patch perl perl-base perl-modules-5.38 pigz
  publicsuffix runc ubuntu-fan xauth xz-utils
```

2. Exec into the container and run docker version.

```
ubuntu@k8s-instance-13:~$ docker exec dind-ubuntu docker version
Client:
 Version:           26.1.3
 API version:       1.45
 Go version:        go1.22.2
 Git commit:        26.1.3-0ubuntu1~24.04.1
 Built:             Mon Oct 14 14:29:26 2024
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
 Engine:
  Version:          28.1.1
  API version:      1.49 (minimum version 1.24)
  Go version:       go1.23.8
  Git commit:       01f442b
  Built:            Fri Apr 18 09:52:14 2025
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.7.27
  GitCommit:        05044ec0a9a75232cad458027ca83437aae3f4da
 runc:
  Version:          1.2.5
  GitCommit:        v1.2.5-0-g59923ef
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

Exercise 7

(Toda en una sola imagen)

1. Run a container with memory and CPU limits:
    a. Memory = 256m
    b. CPU = 0.5 2.
2. Check resource usage stats.
3. Check disk usage (docker system).

```
ubuntu@k8s-instance-13:~$ docker run -d --name limited_container --memory=25
6m --cpus="0.5" alpine sleep 300
47b7eafeddaced4e70da6f2c535e63f3aeae24481f8ae68dab7098388cbc186a
ubuntu@k8s-instance-13:~$ docker stats --no-stream
CONTAINER ID   NAME               CPU %     MEM USAGE / LIMIT     MEM %
 NET I/O          BLOCK I/O     PIDS
47b7eafeddac   limited_container   0.00%     352KiB / 256MiB       0.13%
 446B / 126B      0B / 0B       1
fe622e22337b   dind-ubuntu        0.00%     30.05MiB / 7.749GiB   0.38%
 139MB / 3.1MB    0B / 625MB    1
83d56af09711   bind-test          0.00%     508KiB / 7.749GiB     0.01%
 824B / 126B      0B / 4.1kB    1
52d49bfe54f6   test-volume        0.00%     512KiB / 7.749GiB     0.01%
 950B / 126B      0B / 0B       1
f39daee0a878   nginx2             0.00%     5.754MiB / 7.749GiB   0.07%
 9.85MB / 270kB   0B / 31.4MB   5
5a1703c4b242   nginx1             0.00%     5.863MiB / 7.749GiB   0.07%
 9.85MB / 227kB   0B / 31.4MB   5
96e566817241   nginx-net          0.00%     4.488MiB / 7.749GiB   0.06%
 1.35kB / 126B    0B / 12.3kB   5
d15933aef7c3   ubuntu_container   0.00%     928KiB / 7.749GiB     0.01%
 1.55kB / 126B    0B / 0B       1
04981dc26ed6   nginx-container    0.00%     4.43MiB / 7.749GiB    0.06%
 1.2kB / 126B     0B / 4.1kB    5
ubuntu@k8s-instance-13:~$ docker system df
TYPE            TOTAL     ACTIVE    SIZE      RECLAIMABLE
Images          3         3         278.4MB   0B (0%)
Containers      9         9         551.7MB   0B (0%)
Local Volumes   1         1         63B       0B (0%)
Build Cache     0         0         0B        0B
```

Exercise 8

1. Run a container with policy --restart on-failure. (imagenes conjuntas pa 1 y 2)
2. Kill the container and observe how it restarts.

```
ubuntu@k8s-instance-13:~$ docker run -d --name test_on_failure --restart on-
failure alpine sh -c "exit 1"
eaae6f8295ca2c3e71a758245545fc9121898a82c8a0eaa33139bc98ec329218
ubuntu@k8s-instance-13:~$ sleep 5
ubuntu@k8s-instance-13:~$ docker kill test_on_failure
test_on_failure
```

```
ubuntu@k8s-instance-13:~$ docker logs test_on_failure
ubuntu@k8s-instance-13:~$ docker ps -a | grep test_on_failure
eaae6f8295ca   alpine    "sh -c 'exit 1'"         35 seconds ago      Restar
ting (1) 7 seconds ago                                             test_on_f
ailure
ubuntu@k8s-instance-13:~$ docker ps -a | grep test_on_failure
eaae6f8295ca   alpine    "sh -c 'exit 1'"         56 seconds ago      Restar
ting (1) 2 seconds ago                                             test_on_f
ailure
ubuntu@k8s-instance-13:~$ docker ps -a | grep test_on_failure
eaae6f8295ca   alpine    "sh -c 'exit 1'"         About a minute ago   Exite
d (1) 20 seconds ago                                               test_on_fai
lure
```

3. Try with the policy --restart unless-stopped

```
ubuntu@k8s-instance-13:~$ docker run -d --name test_unless_stopped --restart
 unless-stopped alpine sleep 300
490d4ba7b58f3dd1ab689470f7ec7cbed6e9be4bdbc0747d207e93c521ef44ba
ubuntu@k8s-instance-13:~$ docker ps | grep test_unless_stopped
490d4ba7b58f   alpine    "sleep 300"              11 seconds ago      Up 10
seconds                                                            test_unless_stopped
```

4. Reboot the system and see what happens.

```
ubuntu@k8s-instance-13:~$ sudo reboot

Broadcast message from root@k8s-instance-13 on pts/2 (Fri 2025-05-02 23:41:4
5 -04):

The system will reboot now!
```

```
ubuntu@k8s-instance-13:~$ docker ps | grep test_unless_stopped
490d4ba7b58f   alpine    "sleep 300"   5 minutes ago   Up About a minute
        test_unless_stopped
```

Exercise 9

(Toda en una sola imagen)

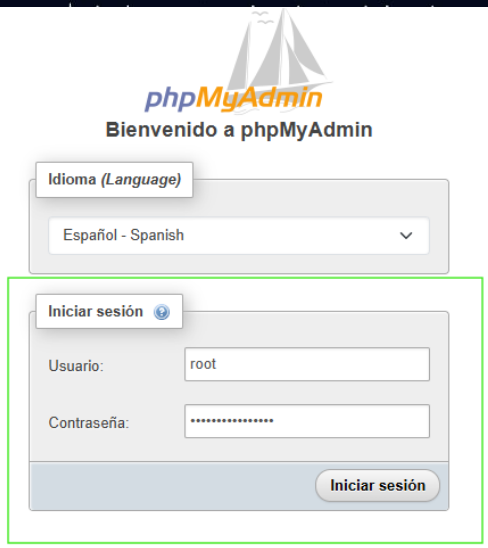
1. Create a network dbnet.
2. Create a volume dbdata.
3. Run a MariaDB container with the following requirements:
    a. Attached to volume dbdata.
    b. Attached to network dbnet.
    c. Do NOT expose ANY port

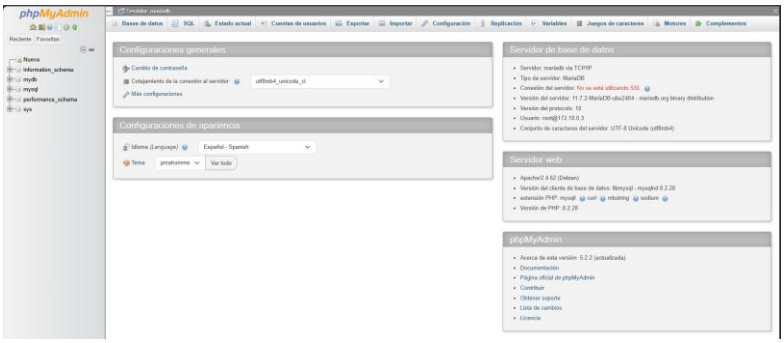
```
ubuntu@k8s-instance-13:~$ docker network create dbnet
3cbd89459f641a186257e25af7e2b9d8db7aae05c636faf434617c703950a654
ubuntu@k8s-instance-13:~$ docker volume create dbdata
dbdata
ubuntu@k8s-instance-13:~$ docker run -d --name mariadb -v dbdata:/var/lib/my
sql --network dbnet -e MARIADB_ROOT_PASSWORD=mysecretpassword -e MARIADB_DAT
ABASE=mydb -e MARIADB_USER=myuser -e MARIADB_PASSWORD=mypassword mariadb:lat
est
Unable to find image 'mariadb:latest' locally
latest: Pulling from library/mariadb
2726e237d1a3: Pull complete
0b86886c6aaa: Pull complete
2b221cf763a8: Pull complete
5e4180757702: Pull complete
43028b9f5f8e: Pull complete
bbef7eafa75b: Pull complete
ab732728101f: Pull complete
0c9f57c1bb30: Pull complete
Digest: sha256:81e893032978c4bf8ad43710b7a979774ed90787fa32d199162148ce28fe3
b76
Status: Downloaded newer image for mariadb:latest
46a90399cf58bd6df634abc5ec6d20da0076c6171dc55f3d0f7446db6066052b
```

Exercise 10

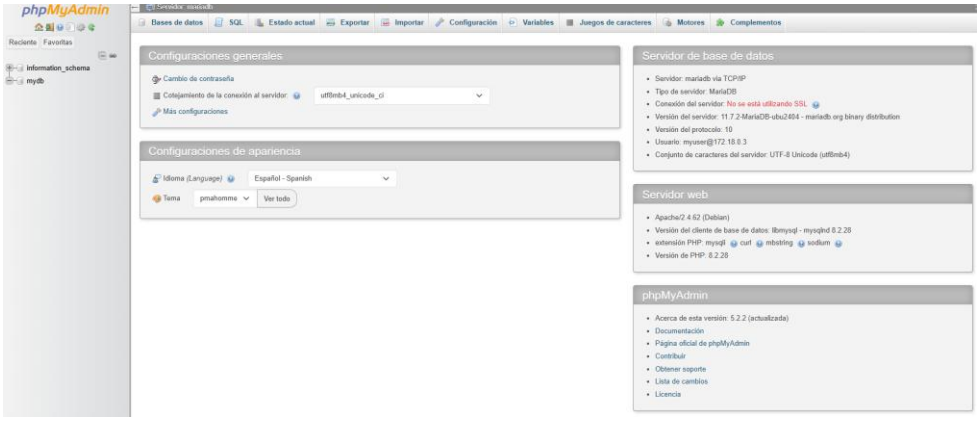
1. Run a PHPMyAdmin container with the following requirements:
    a. Attached to network dbnet (created in Exercise 9).
    b. Use a bind mount to persist the web app configuration.
    c. Linked to the previous MariaDB container (created in Exercise 9)
    d. Open a browser to display the PHPMyAdmin Login Form.
    e. Login with the DB credentials.

```
ubuntu@k8s-instance-13:~$ docker run -d --name phpmyadmin --network dbnet -v
 phpmyadmin_config:/etc/phpmyadmin -e PMA_HOST=mariadb -p 8080:80 phpmyadmin
:latest
a1ce0ee77c57877eb585acf02145aeddeab716e354abd7ec15954cd50f1692c2
```






phpMyAdmin
Bienvenido a phpMyAdmin
Idioma (Language)
Español - Spanish
Iniciar sesión
Usuario:
myuser
Contraseña:
Iniciar sesión


phpMyAdmin
Bases de datos
SQL
Estado actual
Exportar
Importar
Configuración
Variables
Juegos de caracteres
Motores
Complementos
Configuraciones generales
Cambio de contraseña
Más configuraciones
Configuraciones de apariencia
Idioma (Language)
Español - Spanish
Tema
Ver todo
Servidor de base de datos
Servidor web
phpMyAdmin
Acerca de esta versión
Documentación
Página oficial de phpMyAdmin
Contribuir
Obtener soporte
Lista de cambios
Licencia