Docker Networking Experiments

1. Run Containers in Default Bridge Network

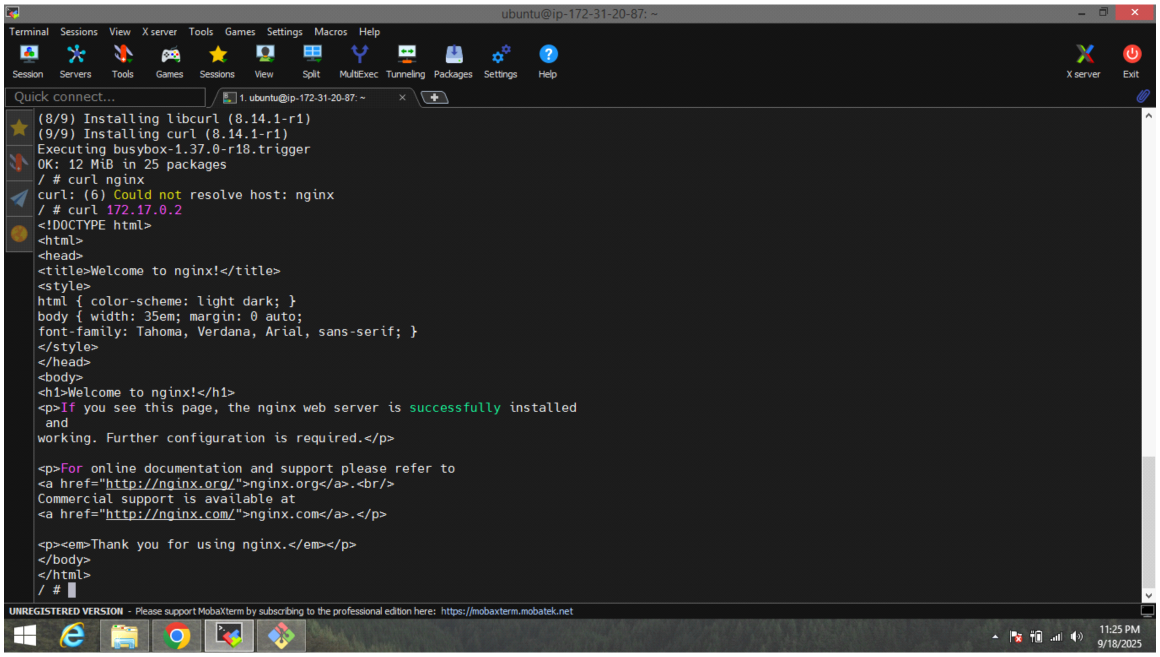
By default, containers connect to the `bridge` network.

Run Nginx container

docker run -d --name nginx-container nginx

Run Alpine container with shell

docker run -it --name alpine-container alpine sh



```
(8/9) Installing libcurl (8.14.1-r1)
(9/9) Installing curl (8.14.1-r1)
Executing busybox-1.37.0-r18.trigger
OK: 12 MiB in 25 packages
/ # curl nginx
curl: (6) Could not resolve host: nginx
/ # curl 172.17.0.2
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed
 and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ #
```

👉 Containers can talk via **IP address**, but not via names.

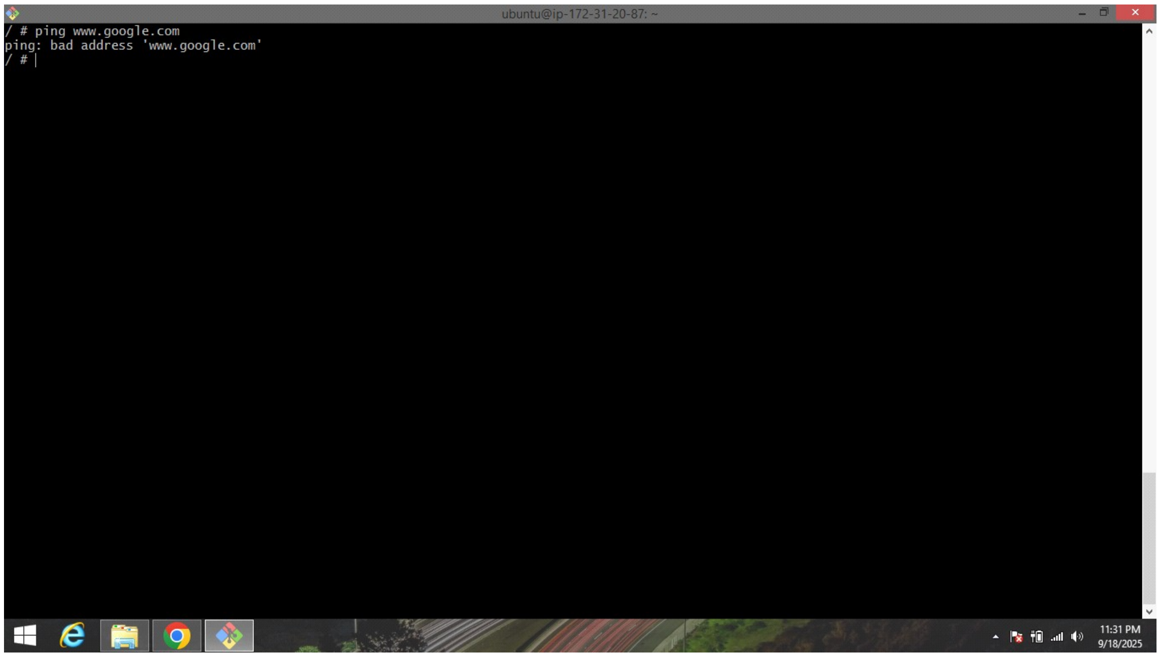
2. Run a Container With No Network

Start a container without any network attached:

docker run -it --name no-net-container --network none alpine sh

No inbound or outbound communication is possible.

Useful for isolated workloads or security restrictions.



```
/ # ping www.google.com
ping: bad address 'www.google.com'
/ # |
```

3. Use a Custom Bridge Network (WordPress + MySQL)

Custom bridge networks allow DNS-based service discovery.

Create a custom bridge network

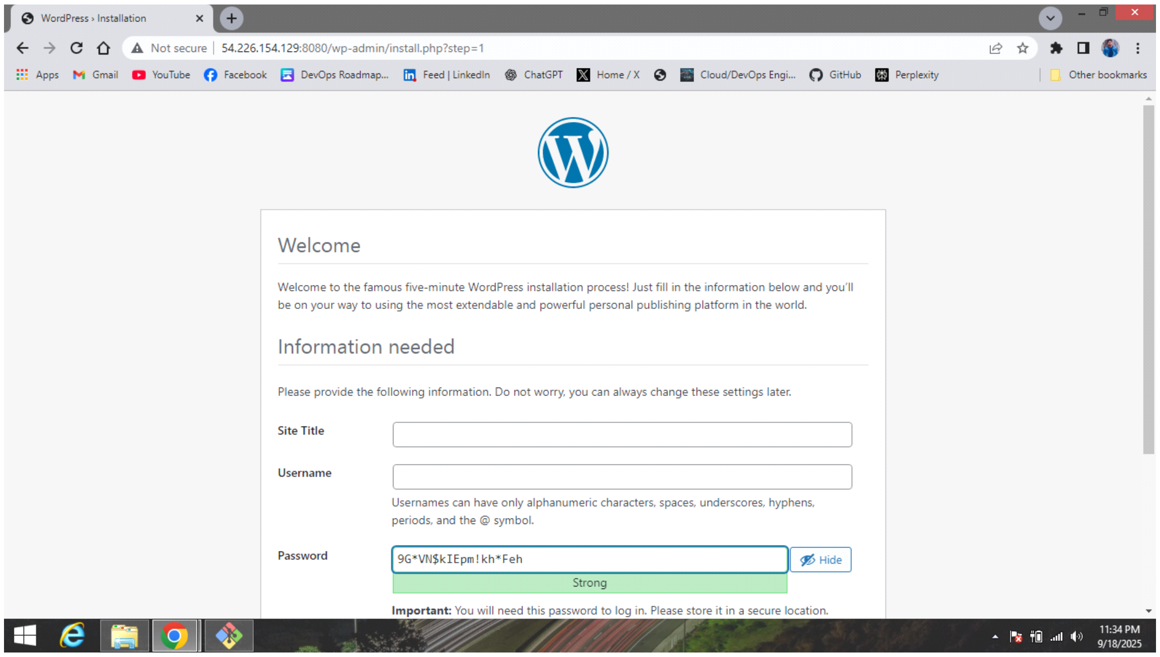
docker network create wp-net

Run MySQL container (hostname = db)

```
docker run -d --name db --network wp-net \
  -e MYSQL_ROOT_PASSWORD=rootpass \
  -e MYSQL_DATABASE=wordpress \
  -e MYSQL_USER=wpuser \
  -e MYSQL_PASSWORD=wppass \
  mysql:8
```

Run WordPress container

```
docker run -d --name wordpress --network wp-net \
  -e WORDPRESS_DB_HOST=db:3306 \
  -e WORDPRESS_DB_USER=wpuser \
  -e WORDPRESS_DB_PASSWORD=wppass \
  -e WORDPRESS_DB_NAME=wordpress \
  -p 8080:80 \
  wordpress:latest
```


WordPress › Installation
Not secure | 54.226.154.129:8080/wp-admin/install.php?step=1
Apps Gmail YouTube Facebook DevOps Roadmap... Feed | LinkedIn ChatGPT Home / X Cloud/DevOps Engi... GitHub Perplexity Other bookmarks
11:34 PM 9/18/2025

Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Do not worry, you can always change these settings later.

Site Title

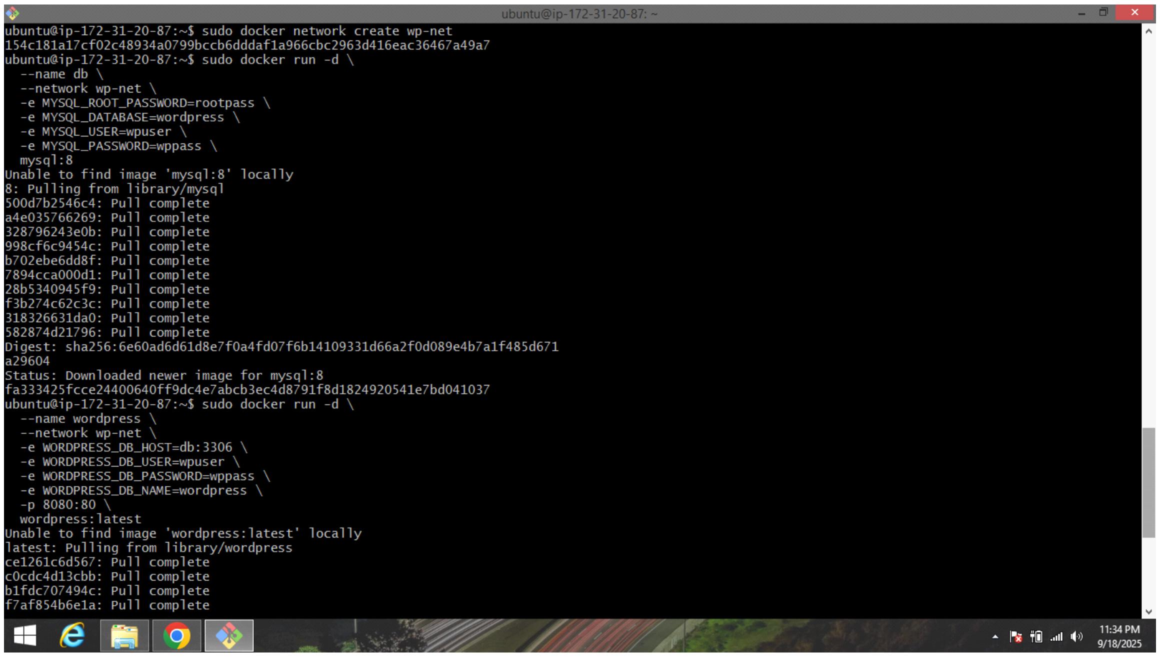
Username

Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password: 9G*VN$kIEpm!kh*Feh   Hide

Strong

Important: You will need this password to log in. Please store it in a secure location.



```
ubuntu@ip-172-31-20-87:~$ sudo docker network create wp-net
154c181a17cf02c48934a0799bccb6dddaf1a966cbc2963d416eac36467a49a7
ubuntu@ip-172-31-20-87:~$ sudo docker run -d \
  --name db \
  --network wp-net \
  -e MYSQL_ROOT_PASSWORD=rootpass \
  -e MYSQL_DATABASE=wordpress \
  -e MYSQL_USER=wpuser \
  -e MYSQL_PASSWORD=wppass \
  mysql:8
Unable to find image 'mysql:8' locally
8: Pulling from library/mysql
500d7b2546c4: Pull complete
a4e035766269: Pull complete
328796243e0b: Pull complete
998cf6c9454c: Pull complete
b702ebe6dd8f: Pull complete
7894cca000d1: Pull complete
28b5340945f9: Pull complete
f3b274c62c3c: Pull complete
318326631da0: Pull complete
582874d21796: Pull complete
Digest: sha256:6e60ad6d61d8e7f0a4fd07f6b14109331d66a2f0d089e4b7a1f485d671
a29604
Status: Downloaded newer image for mysql:8
fa333425fcce24400640ff9dc4e7abcb3ec4d8791f8d1824920541e7bd041037
ubuntu@ip-172-31-20-87:~$ sudo docker run -d \
  --name wordpress \
  --network wp-net \
  -e WORDPRESS_DB_HOST=db:3306 \
  -e WORDPRESS_DB_USER=wpuser \
  -e WORDPRESS_DB_PASSWORD=wppass \
  -e WORDPRESS_DB_NAME=wordpress \
  -p 8080:80 \
  wordpress:latest
Unable to find image 'wordpress:latest' locally
latest: Pulling from library/wordpress
ce1261c6d567: Pull complete
c0cdc4d13cbb: Pull complete
b1fdc707494c: Pull complete
f7af854b6e1a: Pull complete
```

4. Connect a Container to Multiple Networks

A container can belong to more than one network.

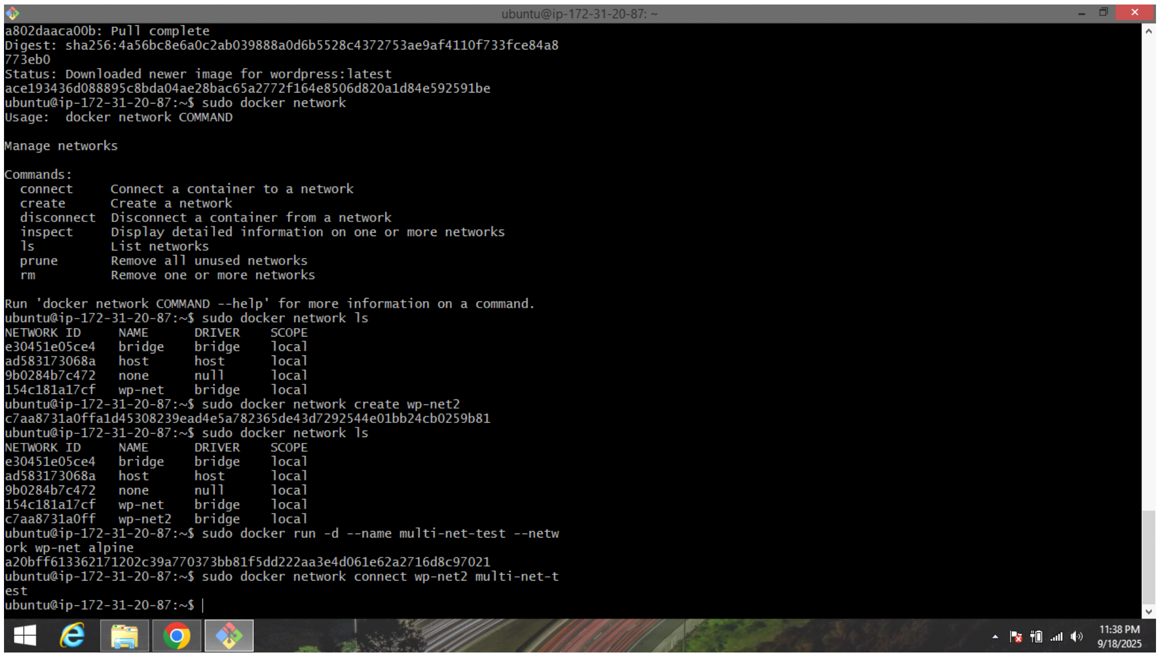
Create two networks

docker network create wp-net

docker network create wp-net2

Start container in wp-net

docker run -d --name multi-net-test --network net1 alpine

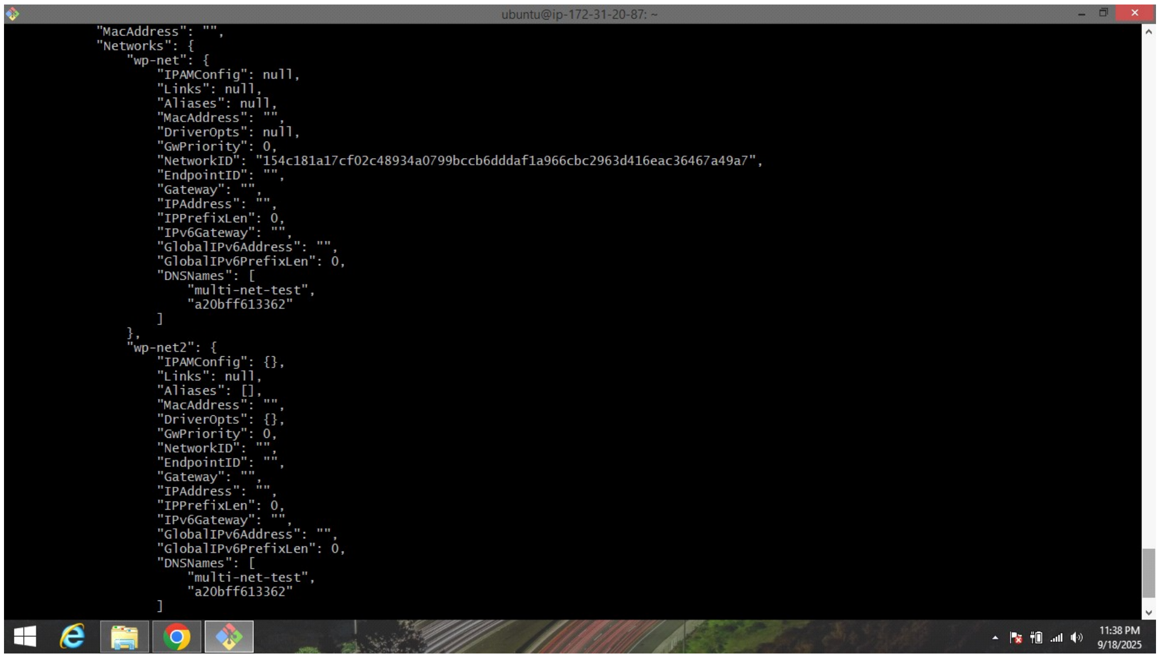


```
a802daaca00b: Pull complete
Digest: sha256:4a56bc8e6a0c2ab039888a0d6b5528c4372753ae9af4110f733fce84a8
773eb0
Status: Downloaded newer image for wordpress:latest
ace193436d088895c8bda04ae28bac65a2772f164e8506d820a1d84e592591be
ubuntu@ip-172-31-20-87:~$ sudo docker network
Usage:  docker network COMMAND

Manage networks

Commands:
  connect     Connect a container to a network
  create      Create a network
  disconnect  Disconnect a container from a network
  inspect     Display detailed information on one or more networks
  ls          List networks
  prune       Remove all unused networks
  rm          Remove one or more networks

Run 'docker network COMMAND --help' for more information on a command.
ubuntu@ip-172-31-20-87:~$ sudo docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
e30451e05ce4   bridge    bridge    local
ad583173068a   host      host      local
9b0284b7c472   none      null      local
154c181a17cf   wp-net    bridge    local
ubuntu@ip-172-31-20-87:~$ sudo docker network create wp-net2
c7aa8731a0ffa1d45308239ead4e5a782365de43d7292544e01bb24cb0259b81
ubuntu@ip-172-31-20-87:~$ sudo docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
e30451e05ce4   bridge    bridge    local
ad583173068a   host      host      local
9b0284b7c472   none      null      local
154c181a17cf   wp-net    bridge    local
c7aa8731a0ff   wp-net2   bridge    local
ubuntu@ip-172-31-20-87:~$ sudo docker run -d --name multi-net-test --netw
ork wp-net alpine
a20bff613362171202c39a770373bb81f5dd222aa3e4d061e62a2716d8c97021
ubuntu@ip-172-31-20-87:~$ sudo docker network connect wp-net2 multi-net-t
est
ubuntu@ip-172-31-20-87:~$ |
```



```
            "MacAddress": "",
            "Networks": {
                "wp-net": {
                    "IPAMConfig": null,
                    "Links": null,
                    "Aliases": null,
                    "MacAddress": "",
                    "DriverOpts": null,
                    "GwPriority": 0,
                    "NetworkID": "154c181a17cf02c48934a0799bccb6dddaf1a966cbc2963d416eac36467a49a7",
                    "EndpointID": "",
                    "Gateway": "",
                    "IPAddress": "",
                    "IPPrefixLen": 0,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "DNSNames": [
                        "multi-net-test",
                        "a20bff613362"
                    ]
                },
                "wp-net2": {
                    "IPAMConfig": {},
                    "Links": null,
                    "Aliases": [],
                    "MacAddress": "",
                    "DriverOpts": {},
                    "GwPriority": 0,
                    "NetworkID": "",
                    "EndpointID": "",
                    "Gateway": "",
                    "IPAddress": "",
                    "IPPrefixLen": 0,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "DNSNames": [
                        "multi-net-test",
                        "a20bff613362"
                    ]
```

Attach container to wp-net2

docker network connect wp-net2 multi-net-test

Inspect container

docker inspect multi-net-test

The container will now have two IPs: one in `wp-net` and another in `wp-net2`.

It can communicate with containers in both networks.

Lessons Learned

Default bridge: IP-based communication only.

None: Completely isolated container.

Custom bridge: DNS-based service discovery by container name.

Multi-network containers: Containers can have multiple IPs and access different networks.