

NIPUN SINGAL
500069052
R171218069
DevOps-III

Application Containerization Lab

EXPERIMENT-4

Create the network and connect with the container

Step1- create the docker network using the command-
docker network create [network-name]

Step2- to list all the network type
docker network ls

Step3- create a container and connect it to the network we created above

**docker run -d --name=[container-name] --net=[network-name]
[image-name]**

Step4- To check all running containers

docker ps

The screenshot shows a web browser window with the URL `https://www.katacoda.com/courses/docker/networking-intro`. The page title is "Docker Networks | Katacoda - Mozilla Firefox". The page content includes a "Task: Create Network" section with the instruction "To start with we create the network with our predefined name." and a code block showing `docker network create backend-network`. Below this is a "Task: Connect To Network" section with the instruction "When we launch new containers, we can use the `--net` attribute to assign which network they should be connected to." and a code block showing `docker run -d --name=redis --net=backend-network redis`. On the right side of the page, there is a terminal window titled "Terminal" showing the following commands and output:

```
$ docker network create backend-network
0819e5633fda20393b6dc875829d91c1fbaac4ee91c04ecad0df1225b35a82fa
$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
0819e5633fda        backend-network     bridge              local
40aaccb68f84        bridge             bridge              local
fa054a9af353        host               host                local
f50397115ef2        none               null                local
$ docker run -d --name=redis --net=backend-network redis
5ed79f77e35f61bc1f03033843737e19365194821be63a927158d40ffc946a44
$ docker ps
CONTAINER ID        IMAGE               COMMAND              CREATED              STATUS
5ed79f77e35f        redis              "docker-entrypoint.s..." 7 seconds ago        Up 6 seconds
6379/tcp           redis
```

Step5- Now run another container and connect it to the same network and ping the other container we created earlier

docker run --net=[network-name] [image-name] ping c1 [already created container name]

c1, it is used to ping only once. If not used it will ping infinitely

note: do not use -d, it will not show the output if used

Applications Places Firefox ESR Mon Feb 8 12:57 AM 1 71 %

Docker Networks | Katacoda - Mozilla Firefox

https://www.katacoda.com/courses/docker/networking-intro

O'REILLY Katakoda KATACODA OVERVIEW & SOLUTIONS CLAIM YOUR PROFILE LOG OUT >

Docker Networks

Step 2 of 5

the *resolv.conf* file.

```
docker run --net=backend-network alpine
cat /etc/resolv.conf ↵
```

When containers attempt to access other containers via a well-known name, such as Redis, the DNS server will return the IP address of the correct Container. In this case, the fully qualified name of Redis will be *redis.backend-network*.

```
docker run --net=backend-network alpine
ping -c1 redis ✓
```

CONTINUE

Terminal +

```
$ docker run --net=backend-network alpine env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=d29570e8a70f
HOME=/root
$ docker run --net=backend-network alpine ping -c1 redis
PING redis (172.19.0.2): 56 data bytes
64 bytes from 172.19.0.2: seq=0 ttl=64 time=0.097 ms

--- redis ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.097/0.097/0.097 ms
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS
PORTS              NAMES
5ed79f77e35f        redis              "docker-entrypoint.s..." 40 seconds ago      Up 39 seconds
6379/tcp           redis
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
5ed79f77e35f	redis	"docker-entrypoint.s..."	40 seconds ago	Up 39 seconds

Step6- Create another network

docker network create [network-name]

Step7- To connect an existing container to network use the command

docker network connect [network-name] [image/container name]

Step8- connect the web server with the same network our container is connected. We will also use port mapping in it

docker run -d -p 3000:3000 --net=[network-name] [docker-hub repository]/[image-name]

Step9- we can test the connection using curl command

curl docker:3000

Applications Places Firefox ESR Mon Feb 8 12:58 AM 1 70 %

Docker Networks | Katacoda - Mozilla Firefox

https://www.katacoda.com/courses/docker/networking-intro

O'REILLY Katakoda KATACODA OVERVIEW & SOLUTIONS CLAIM YOUR PROFILE LOG OUT >

Docker Networks

Step 3 of 5

When using the `connect` command it is possible to attach existing containers to the network.

```
docker network connect frontend-network redis ✓
```

When we launch the web server, given it's attached to the same network it will be able to communicate with our Redis instance.

```
docker run -d -p 3000:3000 --net=frontend-network katacoda/redis-node-docker-example ✓
```

You can test it using `curl docker:3000` ✓

CONTINUE

```
Terminal +
$ docker network create frontend-network
9e745d69c96b9a4f49f4ecaf41b2b938be4faf75b66788c221cd5e9e6c59e06d
$ docker network connect frontend-network redis
$ docker run -d -p 3000:3000 --net=frontend-network katacoda/redis-node-docker-example
Unable to find image 'katacoda/redis-node-docker-example:latest' locally
latest: Pulling from katacoda/redis-node-docker-example
12b41071e6ce: Pull complete
a3ed95caeb02: Pull complete
49a025abf7e3: Pull complete
1fb1c0be01ab: Pull complete
ae8c1f781cde: Pull complete
db73207ad2ae: Pull complete
446b13034c13: Pull complete
Digest: sha256:1aae9759464f00953c8e078a0e0d0649622fef9dd5655b1491f9ee589ae904b4
Status: Downloaded newer image for katacoda/redis-node-docker-example:latest
9fd16885d5f7c8b44ec7c19178d8709ae05e8daee86302424c8528e678791b68
$ curl docker:3000
This page was generated after talking to redis.

Application Build: 1

Total requests: 1

IP count:
::ffff:172.17.0.20: 1
$
```

Step10- to check the networks connected to a container or see the container details, we need to inspect the container

docker inspect [container-name]

The screenshot shows a web browser window with the URL `https://www.katacoda.com/courses/docker/networking-intro`. The page title is "Docker Networks | Katacoda - Mozilla Firefox". The page content includes a "Terminal" window with the following JSON output:

```
{
  "IPPrefixLen": 0,
  "IPv6Gateway": "",
  "MacAddress": "",
  "Networks": {
    "backend-network": {
      "IPAMConfig": null,
      "Links": null,
      "Aliases": [
        "5ed79f77e35f"
      ],
      "NetworkID": "0819e5633fda20393b6dc875829d91c1fbaac4ee91c04ecad0df1225b35a82fa",
      "EndpointID": "67383d7482eec51d021390b3e3b6e84f31081b6ab7ff62af1a400f96151dd3cc",
      "Gateway": "172.19.0.1",
      "IPAddress": "172.19.0.2",
      "IPPrefixLen": 16,
      "IPv6Gateway": "",
      "GlobalIPv6Address": "",
      "GlobalIPv6PrefixLen": 0,
      "MacAddress": "02:42:ac:13:00:02",
      "DriverOpts": null
    },
    "frontend-network": {
      "IPAMConfig": {},
      "Links": null,
      "Aliases": [
        "5ed79f77e35f"
      ],
      "NetworkID": "9e745d69c96b9a4f49f4ecaf41b2b938be4faf75b66788c221cd5e9e6c59e06d",

```

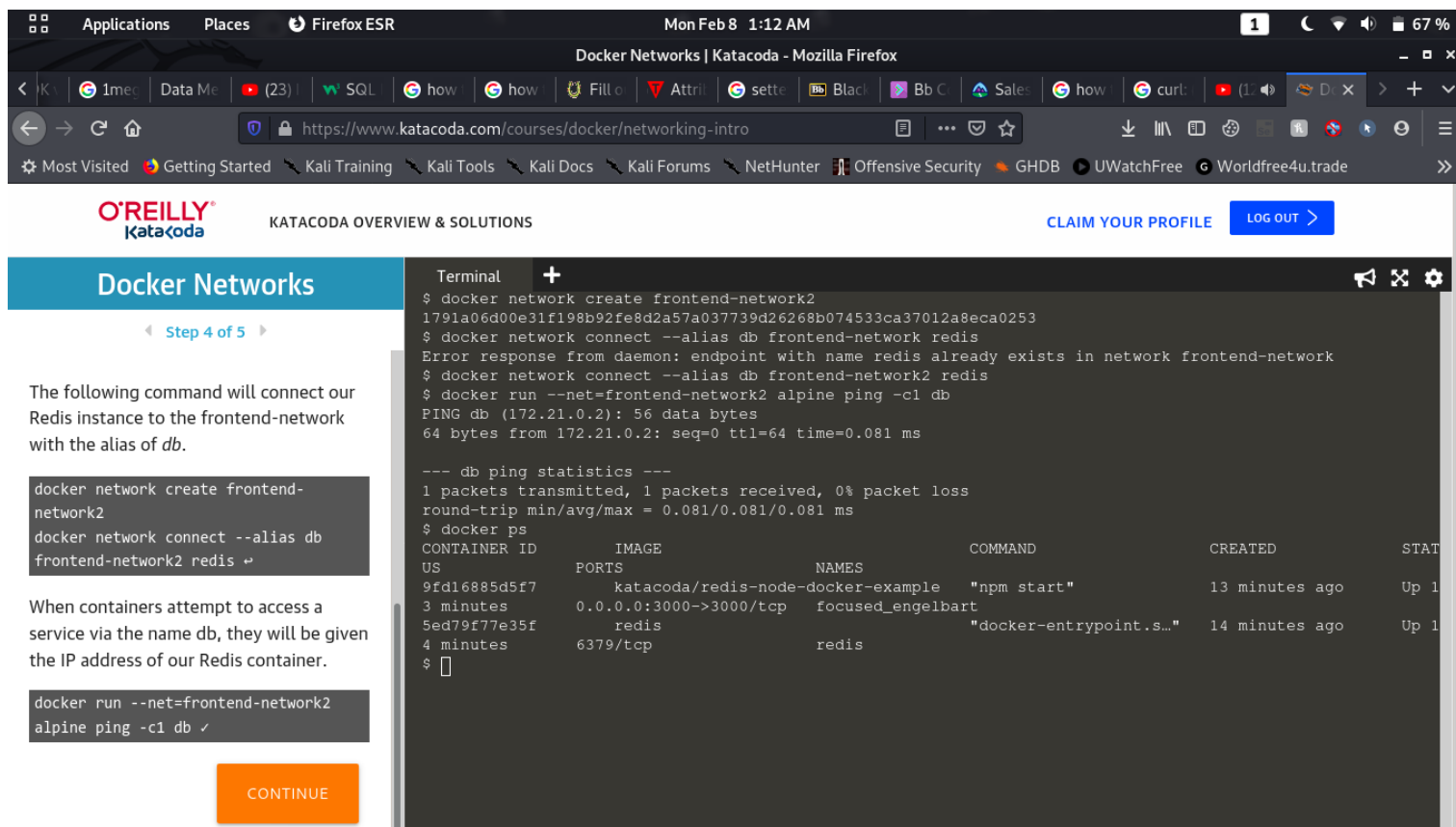
Step11- Create another network and connect that network to a container using alias option. Alias is used to give a name to the container on the network and do not reveal the real name of the container to others on network

docker network create [network-name]

docker network connect --alias [name] [network-name] [container-name]

Step12- Now run a new container, connect to the network created in step11 and ping the container of step11 using alias name

docker run --net=[network-name] [image-name] ping -c1 [alias-name of another container]



The screenshot shows a web browser window displaying a Katakoda course page for 'Docker Networks'. The page includes a terminal window with the following commands and output:

```
$ docker network create frontend-network2
1791a06d00e31f198b92fe8d2a57a037739d26268b074533ca37012a8eca0253
$ docker network connect --alias db frontend-network redis
Error response from daemon: endpoint with name redis already exists in network frontend-network
$ docker network connect --alias db frontend-network2 redis
$ docker run --net=frontend-network2 alpine ping -c1 db
PING db (172.21.0.2): 56 data bytes
64 bytes from 172.21.0.2: seq=0 ttl=64 time=0.081 ms

--- db ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.081/0.081/0.081 ms
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
9fd16885d5f7	katakoda/redis-node-docker-example	"npm start"	13 minutes ago	Up 1
3 minutes	0.0.0.0:3000->3000/tcp	focused_engelbart		
5ed79f77e35f	redis	"docker-entrypoint.s..."	14 minutes ago	Up 1
4 minutes	6379/tcp	redis		

The page also includes instructions on how to connect a Redis instance to the frontend-network and how to ping it from a new container. A 'CONTINUE' button is visible at the bottom.

Step13- we can inspect the network or see network details using

docker network inspect [network-name]

Applications Places Firefox ESR Mon Feb 8 1:22 AM 1 64 %

Docker Networks | Katacoda - Mozilla Firefox

https://www.katacoda.com/courses/docker/networking-intro

O'REILLY Katakoda KATACODA OVERVIEW & SOLUTIONS CLAIM YOUR PROFILE LOG OUT >

Docker Networks

◀ Step 5 of 5 ▶

networks on our host.

```
docker network ls ✓
```

We can then explore the network to see which containers are attached and their IP addresses.

```
docker network inspect frontend-network ✓
```

The following command disconnects the redis container from the *frontend-network*.

```
docker network disconnect frontend-network redis ↵
```

CONTINUE

Terminal +

```
$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
0819e5633fda        backend-network     bridge              local
40aac0b68f84        bridge              bridge              local
9e745d69c96b        frontend-network    bridge              local
1791a06d00e3        frontend-network2   bridge              local
fa054a9af353        host                host                local
f50397115ef2        none                null                local

$ docker network inspect frontend-network
[
  {
    "Name": "frontend-network",
    "Id": "9e745d69c96b9a4f49f4ecaf41b2b938be4faf75b66788c221cd5e9e6c59e06d",
    "Created": "2021-02-07T19:27:17.046765878Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.20.0.0/16",
          "Gateway": "172.20.0.1"
        }
      ]
    }
  },
  {
    "Internal": false,
```

Step14- To disconnect a network from the container we use the command **docker network disconnect [network-name] [container-name]**

Applications Places Firefox ESR Mon Feb 8 1:26 AM 1 63 %

Docker Networks | Katacoda - Mozilla Firefox

https://www.katacoda.com/courses/docker/networking-intro

O'REILLY Katakoda KATACODA OVERVIEW & SOLUTIONS CLAIM YOUR PROFILE LOG OUT >

Docker Networks

◀ Step 5 of 5 ▶

which containers are attached and their IP addresses.

```
docker network inspect frontend-network ✓
```

The following command disconnects the redis container from the *frontend-network*.

```
docker network disconnect frontend-network redis ✓
```

CONTINUE

Terminal +

```
$ docker network disconnect frontend-network redis
$
```