Application Containerization Lab

Aman Kumar Gupta

CSE-DevOps 18

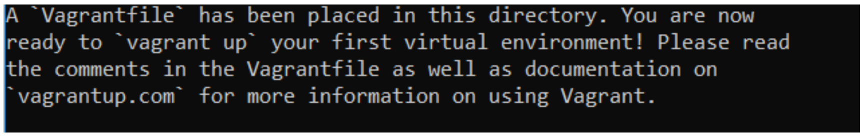
R171218016-500067759

Experiment 1

AIM: Install Vagrant and create basic vagrant box using Virtual Box.

Download and Install Oracle Virtual Box and Vagrant.

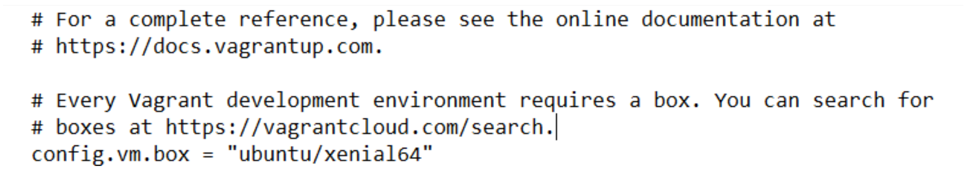
Run vagrant init command



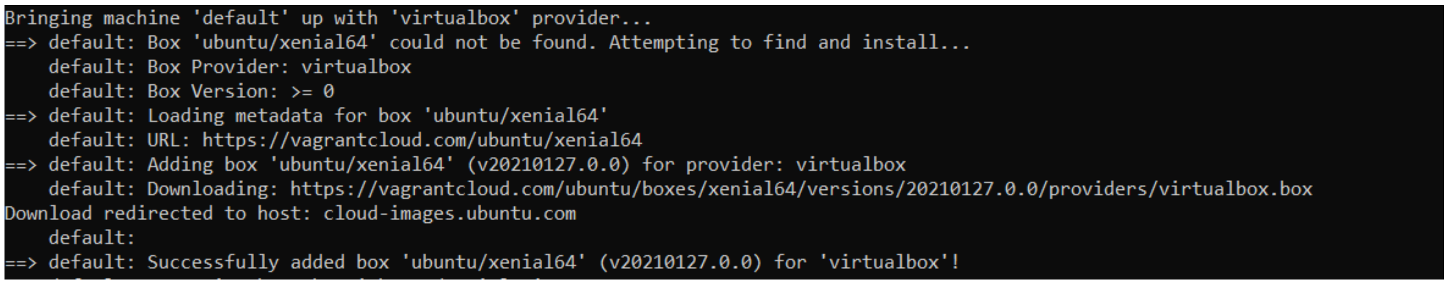
```
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
```

Editing “Vagrantfile” and adding ubuntu/xenial64 in box tag.

Run vagrant up command

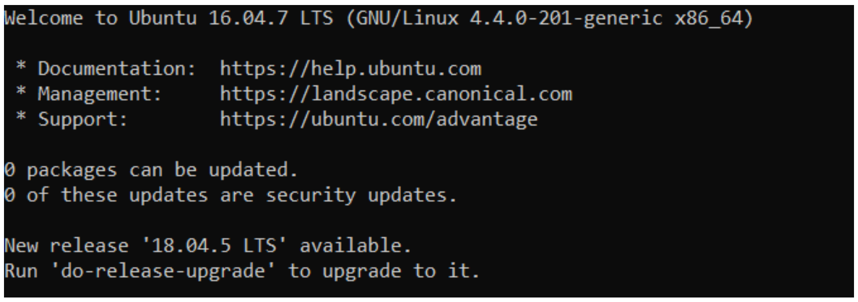


```
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You can search for
  # boxes at https://vagrantcloud.com/search.
  config.vm.box = "ubuntu/xenial64"
```



```
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Box 'ubuntu/xenial64' could not be found. Attempting to find and install...
    default: Box Provider: virtualbox
    default: Box Version: >= 0
==> default: Loading metadata for box 'ubuntu/xenial64'
    default: URL: https://vagrantcloud.com/ubuntu/xenial64
==> default: Adding box 'ubuntu/xenial64' (v20210127.0.0) for provider: virtualbox
    default: Downloading: https://vagrantcloud.com/ubuntu/boxes/xenial64/versions/20210127.0.0/providers/virtualbox.box
Download redirected to host: cloud-images.ubuntu.com
    default:
==> default: Successfully added box 'ubuntu/xenial64' (v20210127.0.0) for 'virtualbox'!
```

Run vagrant SSH command



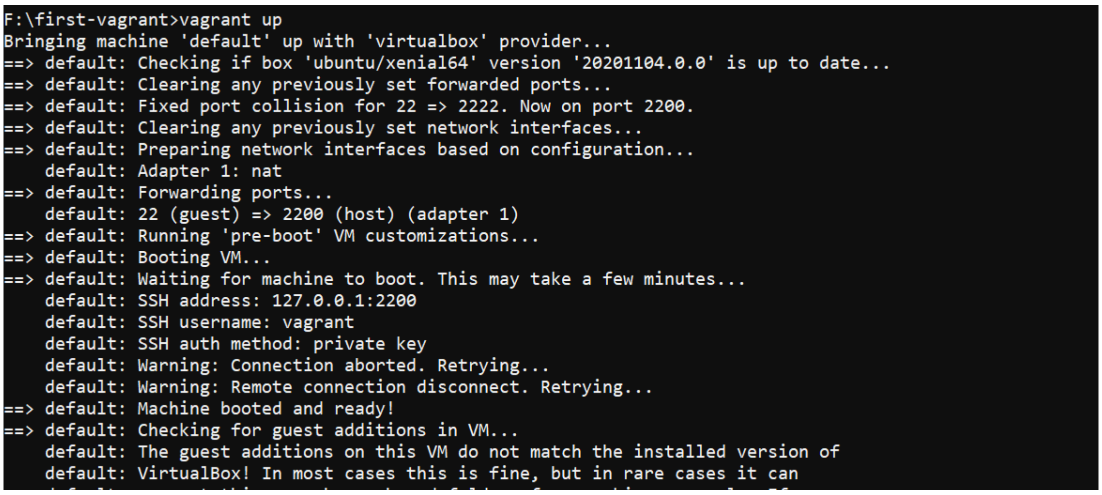
```
Welcome to Ubuntu 16.04.7 LTS (GNU/Linux 4.4.0-201-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 packages can be updated.
0 of these updates are security updates.

New release '18.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
```

Experiment 2

Aim: Install Docker in a VM, create docker volume and share data between containers

Starting VM

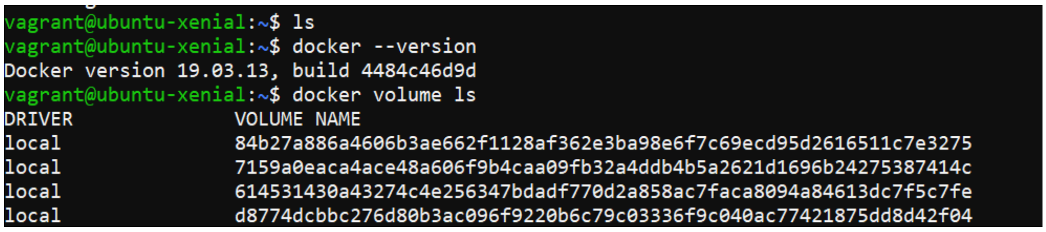
`$ vagrant up`



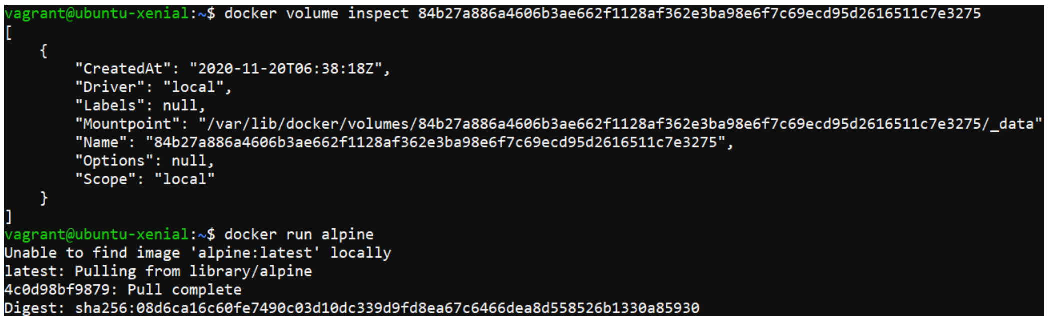
```
F:\first-vagrant>vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'ubuntu/xenial64' version '20201104.0.0' is up to date...
==> default: Clearing any previously set forwarded ports...
==> default: Fixed port collision for 22 => 2222. Now on port 2200.
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2200 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2200
    default: SSH username: vagrant
    default: SSH auth method: private key
    default: Warning: Connection aborted. Retrying...
    default: Warning: Remote connection disconnect. Retrying...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
    default: The guest additions on this VM do not match the installed version of
    default: VirtualBox! In most cases this is fine, but in rare cases it can
```

Listing all the volumes in docker and inspecting a volume

$ docker volume ls



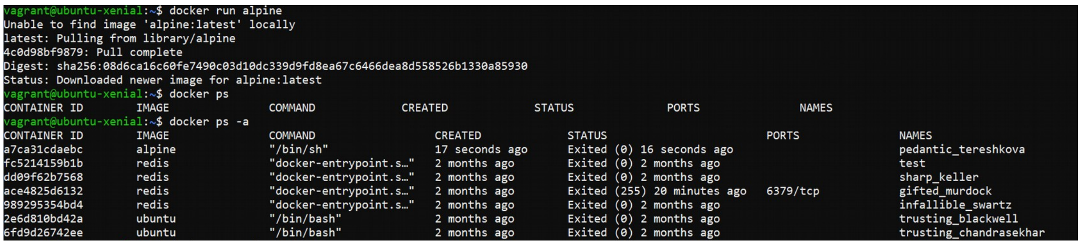
```
vagrant@ubuntu-xenial:~$ ls
vagrant@ubuntu-xenial:~$ docker --version
Docker version 19.03.13, build 4484c46d9d
vagrant@ubuntu-xenial:~$ docker volume ls
DRIVER              VOLUME NAME
local               84b27a886a4606b3ae662f1128af362e3ba98e6f7c69ecd95d2616511c7e3275
local               7159a0eaca4ace48a606f9b4caa09fb32a4ddb4b5a2621d1696b24275387414c
local               614531430a43274c4e256347bdadf770d2a858ac7faca8094a84613dc7f5c7fe
local               d8774dcbbc276d80b3ac096f9220b6c79c03336f9c040ac77421875dd8d42f04
```



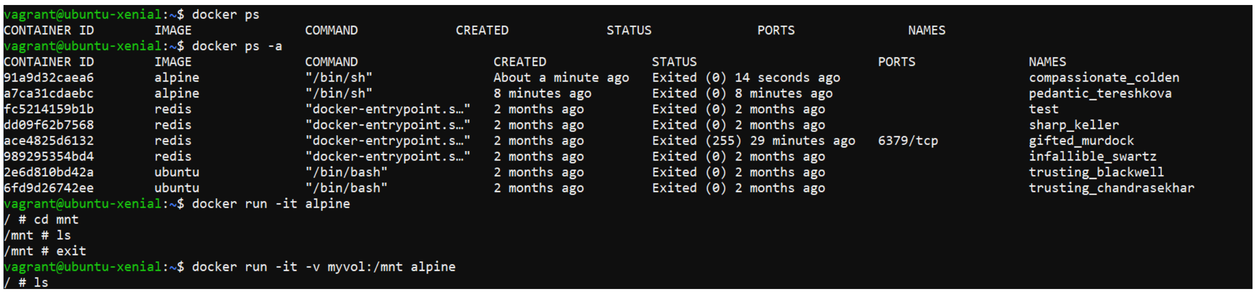
```
vagrant@ubuntu-xenial:~$ docker volume inspect 84b27a886a4606b3ae662f1128af362e3ba98e6f7c69ecd95d2616511c7e3275
[
    {
        "CreatedAt": "2020-11-20T06:38:18Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/84b27a886a4606b3ae662f1128af362e3ba98e6f7c69ecd95d2616511c7e3275/_data"
        "Name": "84b27a886a4606b3ae662f1128af362e3ba98e6f7c69ecd95d2616511c7e3275",
        "Options": null,
        "Scope": "local"
    }
]
vagrant@ubuntu-xenial:~$ docker run alpine
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
4c0d98bf9879: Pull complete
Digest: sha256:08d6ca16c60fe7490c03d10dc339d9fd8ea67c6466dea8d558526b1330a85930
```

Running an Alpine Image

`$ docker run alpine`



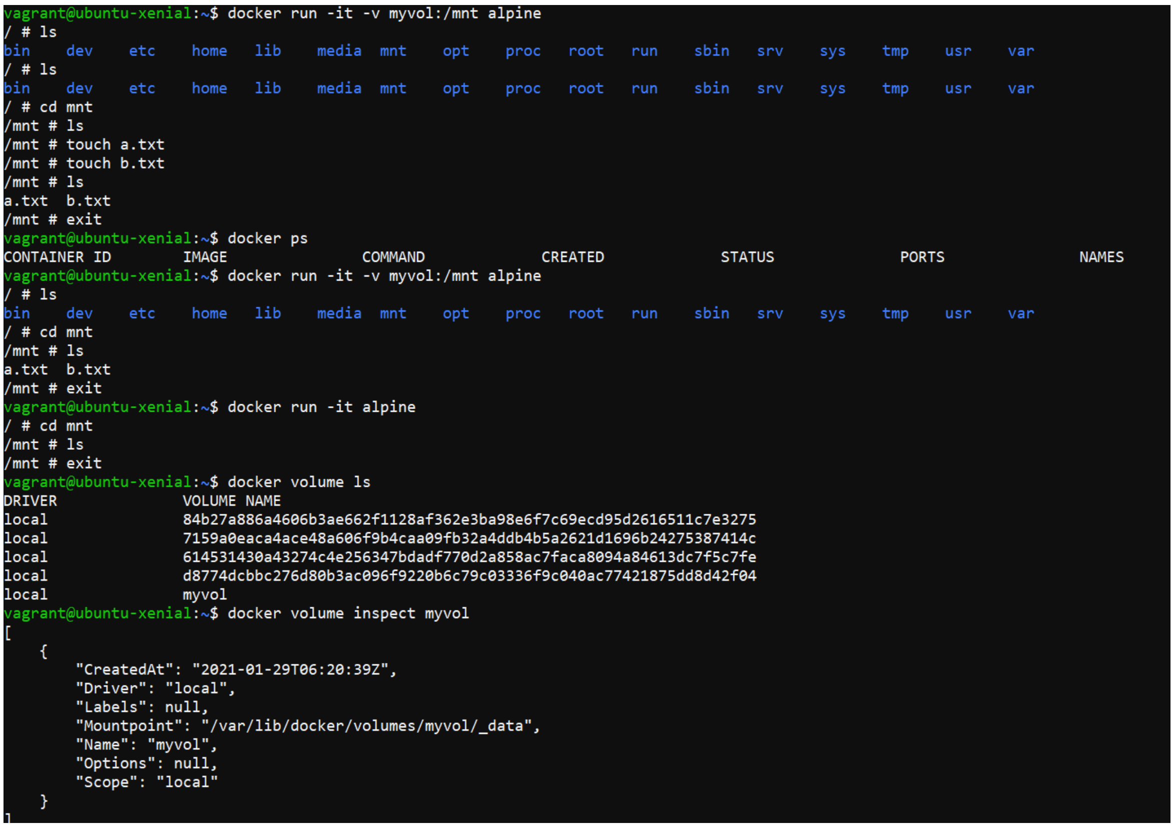
```
vagrant@ubuntu-xenial:~$ docker run alpine
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
4c0d98bf9879: Pull complete
Digest: sha256:08d6ca16c60fe7490c03d10dc339d9fd8ea67c6466dea8d558526b1330a85930
Status: Downloaded newer image for alpine:latest
vagrant@ubuntu-xenial:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
vagrant@ubuntu-xenial:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS               NAMES
a7ca31cdaebc        alpine              "/bin/sh"                17 seconds ago      Exited (0) 16 seconds ago                       pedantic_tereshkova
fc5214159b1b        redis               "docker-entrypoint.s…"   2 months ago        Exited (0) 2 months ago                         test
dd09f62b7568        redis               "docker-entrypoint.s…"   2 months ago        Exited (0) 2 months ago                         sharp_keller
ace4825d6132        redis               "docker-entrypoint.s…"   2 months ago        Exited (255) 20 minutes ago   6379/tcp          gifted_murdock
989295354bd4        redis               "docker-entrypoint.s…"   2 months ago        Exited (0) 2 months ago                         infallible_swartz
2e6d810bd42a        ubuntu              "/bin/bash"              2 months ago        Exited (0) 2 months ago                         trusting_blackwell
6fd9d26742ee        ubuntu              "/bin/bash"              2 months ago        Exited (0) 2 months ago                         trusting_chandrasekhar
```



```
vagrant@ubuntu-xenial:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
vagrant@ubuntu-xenial:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS                        PORTS               NAMES
91a9d32caea6        alpine              "/bin/sh"                About a minute ago   Exited (0) 14 seconds ago                         compassionate_colden
a7ca31cdaebc        alpine              "/bin/sh"                8 minutes ago        Exited (0) 8 minutes ago                          pedantic_tereshkova
fc5214159b1b        redis               "docker-entrypoint.s…"   2 months ago         Exited (0) 2 months ago                           test
dd09f62b7568        redis               "docker-entrypoint.s…"   2 months ago         Exited (0) 2 months ago                           sharp_keller
ace4825d6132        redis               "docker-entrypoint.s…"   2 months ago         Exited (255) 29 minutes ago   6379/tcp            gifted_murdock
989295354bd4        redis               "docker-entrypoint.s…"   2 months ago         Exited (0) 2 months ago                           infallible_swartz
2e6d810bd42a        ubuntu              "/bin/bash"              2 months ago         Exited (0) 2 months ago                           trusting_blackwell
6fd9d26742ee        ubuntu              "/bin/bash"              2 months ago         Exited (0) 2 months ago                           trusting_chandrasekhar
vagrant@ubuntu-xenial:~$ docker run -it alpine
/ # cd mnt
/mnt # ls
/mnt # exit
vagrant@ubuntu-xenial:~$ docker run -it -v myvol:/mnt alpine
/ # ls
```

Creating some files in that alpine cont.

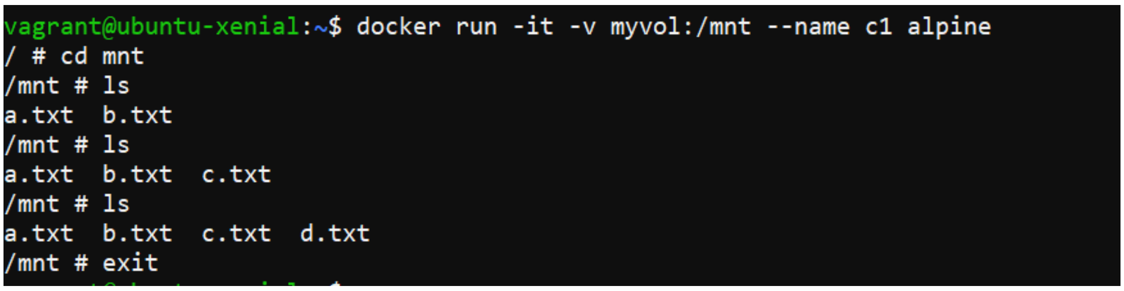
$ docker run –it –v myvol:/mnt alpine



```
vagrant@ubuntu-xenial:~$ docker run -it -v myvol:/mnt alpine
/ # ls
bin    dev    etc    home   lib    media  mnt    opt    proc   root   run    sbin   srv    sys    tmp    usr    var
/ # ls
bin    dev    etc    home   lib    media  mnt    opt    proc   root   run    sbin   srv    sys    tmp    usr    var
/ # cd mnt
/mnt # ls
/mnt # touch a.txt
/mnt # touch b.txt
/mnt # ls
a.txt  b.txt
/mnt # exit
vagrant@ubuntu-xenial:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
vagrant@ubuntu-xenial:~$ docker run -it -v myvol:/mnt alpine
/ # ls
bin    dev    etc    home   lib    media  mnt    opt    proc   root   run    sbin   srv    sys    tmp    usr    var
/ # cd mnt
/mnt # ls
a.txt  b.txt
/mnt # exit
vagrant@ubuntu-xenial:~$ docker run -it alpine
/ # cd mnt
/mnt # ls
/mnt # exit
vagrant@ubuntu-xenial:~$ docker volume ls
DRIVER              VOLUME NAME
local               84b27a886a4606b3ae662f1128af362e3ba98e6f7c69ecd95d2616511c7e3275
local               7159a0eaca4ace48a606f9b4caa09fb32a4ddb4b5a2621d1696b24275387414c
local               614531430a43274c4e256347bdadf770d2a858ac7faca8094a84613dc7f5c7fe
local               d8774dcbbc276d80b3ac096f9220b6c79c03336f9c040ac77421875dd8d42f04
local               myvol
vagrant@ubuntu-xenial:~$ docker volume inspect myvol
[
    {
        "CreatedAt": "2021-01-29T06:20:39Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/myvol/_data",
        "Name": "myvol",
        "Options": null,
        "Scope": "local"
    }
]
```

Creating Container with Volume

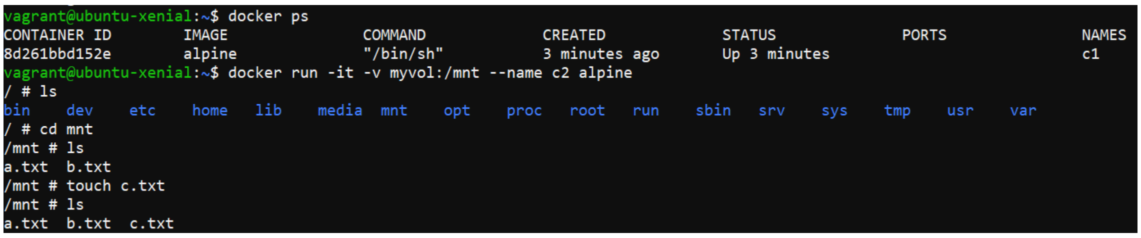
$ docker run –it –v myvol:/mnt –name c1 alpine



```
vagrant@ubuntu-xenial:~$ docker run -it -v myvol:/mnt --name c1 alpine
/ # cd mnt
/mnt # ls
a.txt  b.txt
/mnt # ls
a.txt  b.txt  c.txt
/mnt # ls
a.txt  b.txt  c.txt  d.txt
/mnt # exit
```

In another instance of terminal, starting a VM and running docker in that

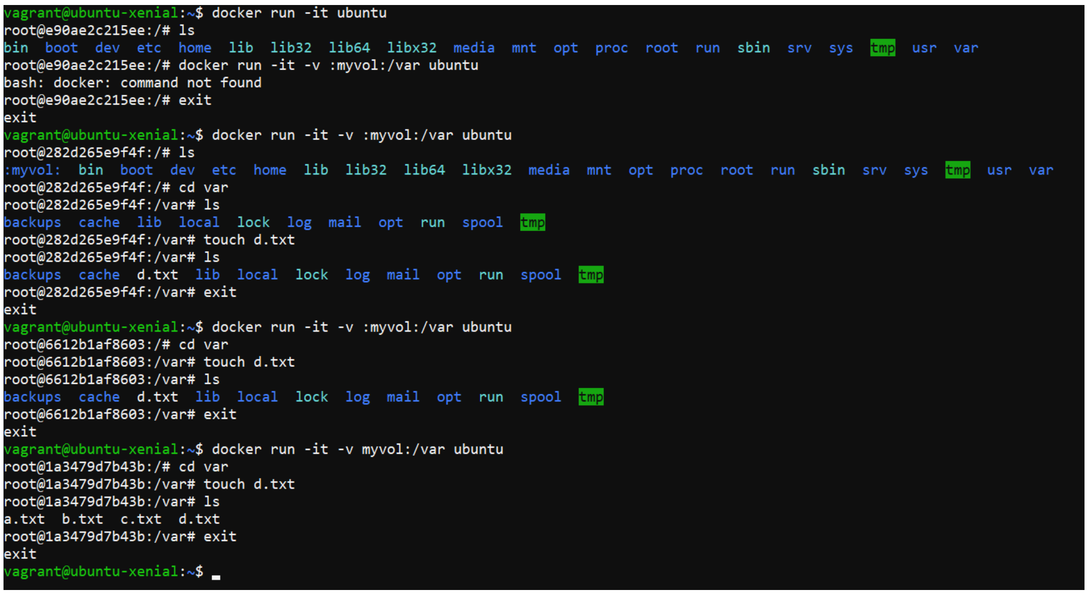
`$ docker run –it –v myvol:/mnt –name c2 alpine`



```
vagrant@ubuntu-xenial:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
8d261bbd152e        alpine              "/bin/sh"           3 minutes ago       Up 3 minutes                            c1
vagrant@ubuntu-xenial:~$ docker run -it -v myvol:/mnt --name c2 alpine
/ # ls
bin    dev    etc    home   lib    media  mnt    opt    proc   root   run    sbin   srv    sys    tmp    usr    var
/ # cd mnt
/mnt # ls
a.txt  b.txt
/mnt # touch c.txt
/mnt # ls
a.txt  b.txt  c.txt
```

All the files show synced up

$ docker run -it –v myvol:/var ubuntu



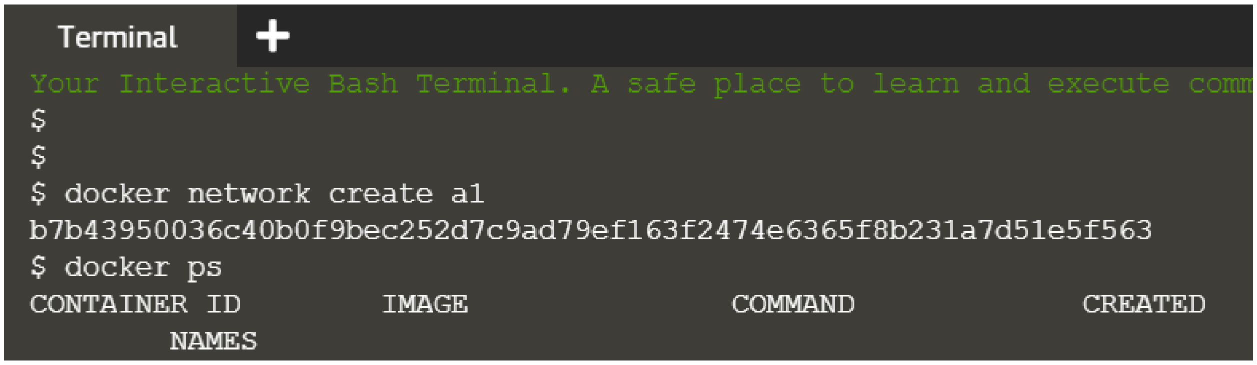
```
vagrant@ubuntu-xenial:~$ docker run -it ubuntu
root@e90ae2c215ee:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@e90ae2c215ee:/# docker run -it -v :myvol:/var ubuntu
bash: docker: command not found
root@e90ae2c215ee:/# exit
exit
vagrant@ubuntu-xenial:~$ docker run -it -v :myvol:/var ubuntu
root@282d265e9f4f:/# ls
:myvol:  bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@282d265e9f4f:/# cd var
root@282d265e9f4f:/var# ls
backups  cache  lib  local  lock  log  mail  opt  run  spool  tmp
root@282d265e9f4f:/var# touch d.txt
root@282d265e9f4f:/var# ls
backups  cache  d.txt  lib  local  lock  log  mail  opt  run  spool  tmp
root@282d265e9f4f:/var# exit
exit
vagrant@ubuntu-xenial:~$ docker run -it -v :myvol:/var ubuntu
root@6612b1af8603:/# cd var
root@6612b1af8603:/var# touch d.txt
root@6612b1af8603:/var# ls
backups  cache  d.txt  lib  local  lock  log  mail  opt  run  spool  tmp
root@6612b1af8603:/var# exit
exit
vagrant@ubuntu-xenial:~$ docker run -it -v myvol:/var ubuntu
root@1a3479d7b43b:/# cd var
root@1a3479d7b43b:/var# touch d.txt
root@1a3479d7b43b:/var# ls
a.txt  b.txt  c.txt  d.txt
root@1a3479d7b43b:/var# exit
exit
vagrant@ubuntu-xenial:~$ _
```

Experiment 3

Aim: Create network and attach multiple containers to it.

Check if any container is running

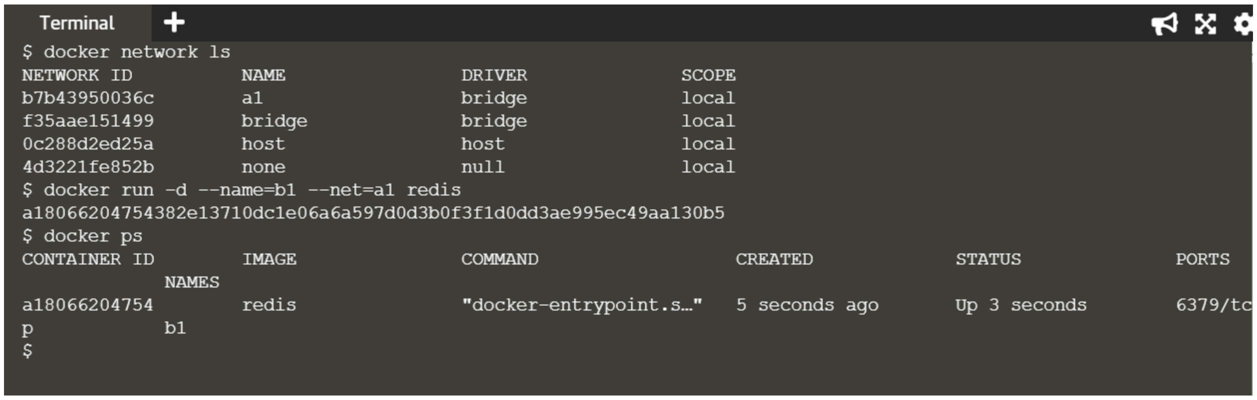
Run command – `$ docker ps`



```
Terminal  +
Your Interactive Bash Terminal. A safe place to learn and execute comm
$
$
$ docker network create a1
b7b43950036c40b0f9bec252d7c9ad79ef163f2474e6365f8b231a7d51e5f563
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
        NAMES
```

Creating a container

Run command – `$ docker run –d –name=b1 –net=a1 redis`

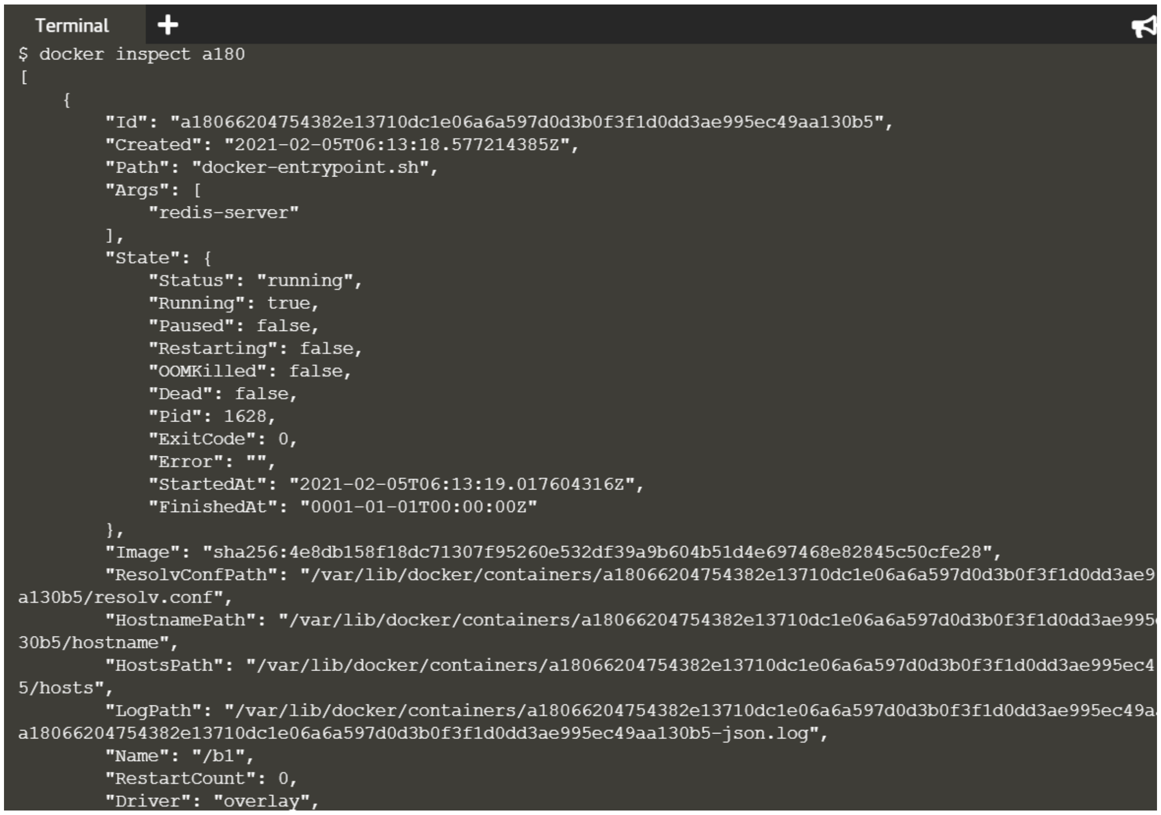


```
$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
b7b43950036c        a1                  bridge              local
f35aae151499        bridge              bridge              local
0c288d2ed25a        host                host                local
4d3221fe852b        none                null                local
$ docker run -d --name=b1 --net=a1 redis
a18066204754382e13710dc1e06a6a597d0d3b0f3f1d0dd3ae995ec49aa130b5
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
      NAMES
a18066204754        redis               "docker-entrypoint.s…"   5 seconds ago       Up 3 seconds        6379/tc
p                   b1
$
```

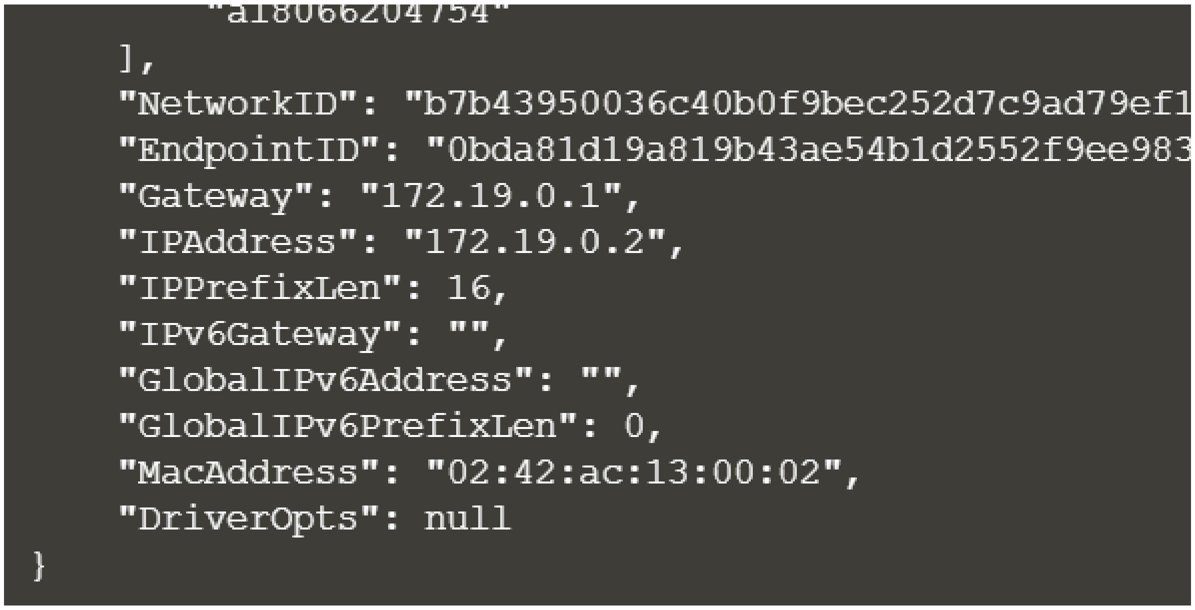
To check IP address

Run command - `$ docker inspect a180`

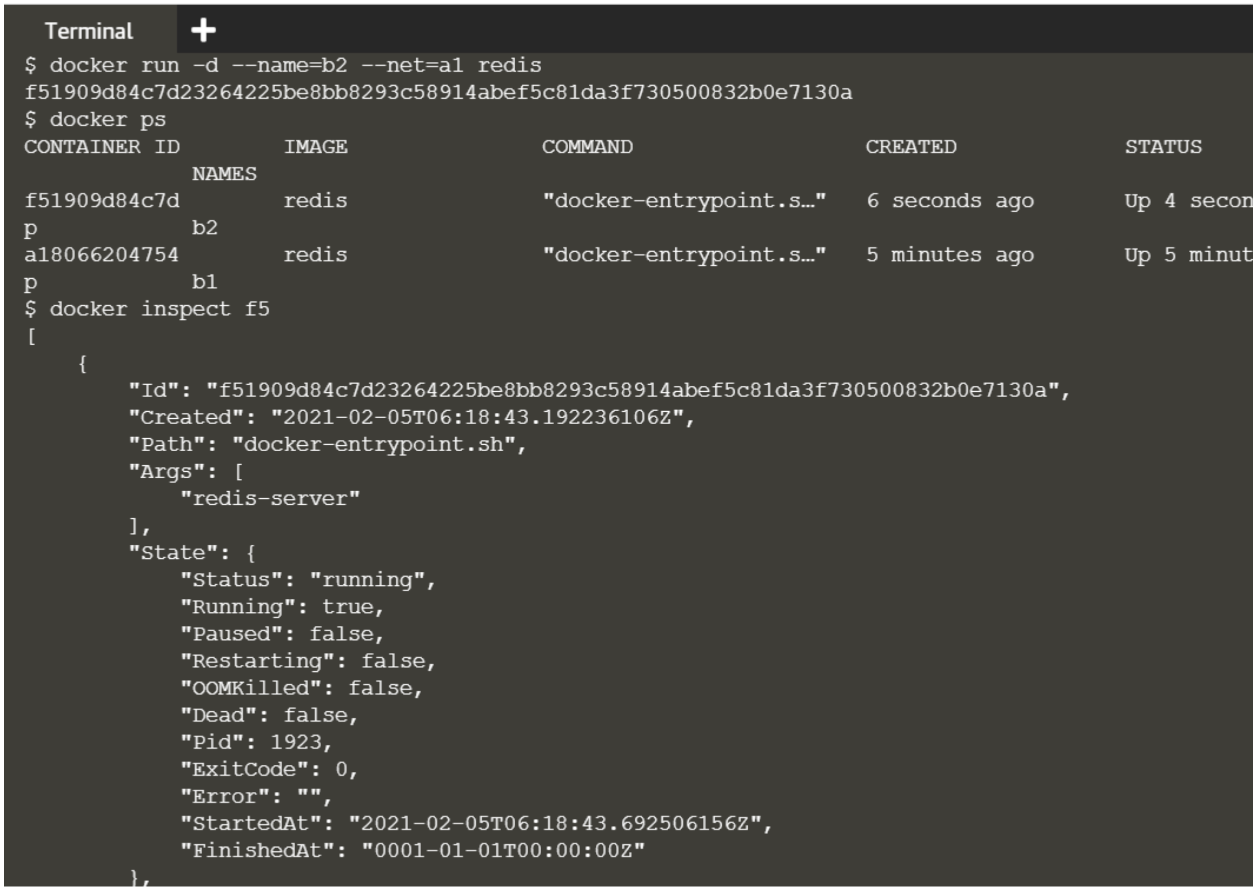
(Note : a180 is unique initials of container ID you want ip)


Terminal

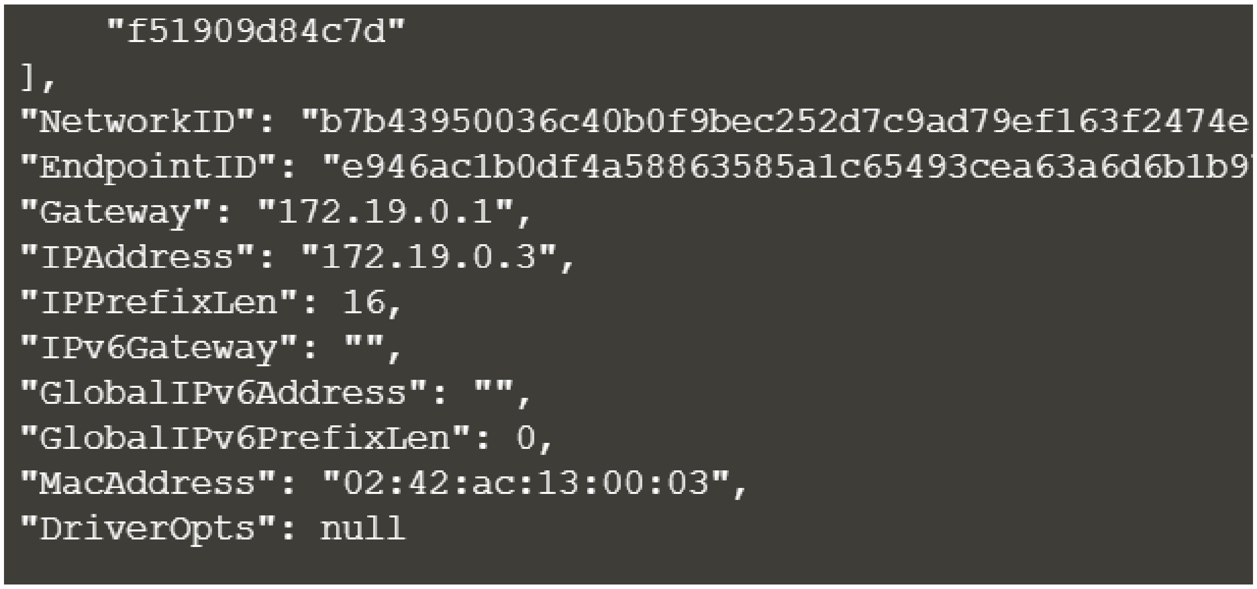
```
$ docker inspect a180
[
    {
        "Id": "a18066204754382e13710dc1e06a6a597d0d3b0f3f1d0dd3ae995ec49aa130b5",
        "Created": "2021-02-05T06:13:18.577214385Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "redis-server"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 1628,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2021-02-05T06:13:19.017604316Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
        "Image": "sha256:4e8db158f18dc71307f95260e532df39a9b604b51d4e697468e82845c50cfe28",
        "ResolvConfPath": "/var/lib/docker/containers/a18066204754382e13710dc1e06a6a597d0d3b0f3f1d0dd3ae9
a130b5/resolv.conf",
        "HostnamePath": "/var/lib/docker/containers/a18066204754382e13710dc1e06a6a597d0d3b0f3f1d0dd3ae995
30b5/hostname",
        "HostsPath": "/var/lib/docker/containers/a18066204754382e13710dc1e06a6a597d0d3b0f3f1d0dd3ae995ec4
5/hosts",
        "LogPath": "/var/lib/docker/containers/a18066204754382e13710dc1e06a6a597d0d3b0f3f1d0dd3ae995ec49a
a18066204754382e13710dc1e06a6a597d0d3b0f3f1d0dd3ae995ec49aa130b5-json.log",
        "Name": "/b1",
        "RestartCount": 0,
        "Driver": "overlay",
```



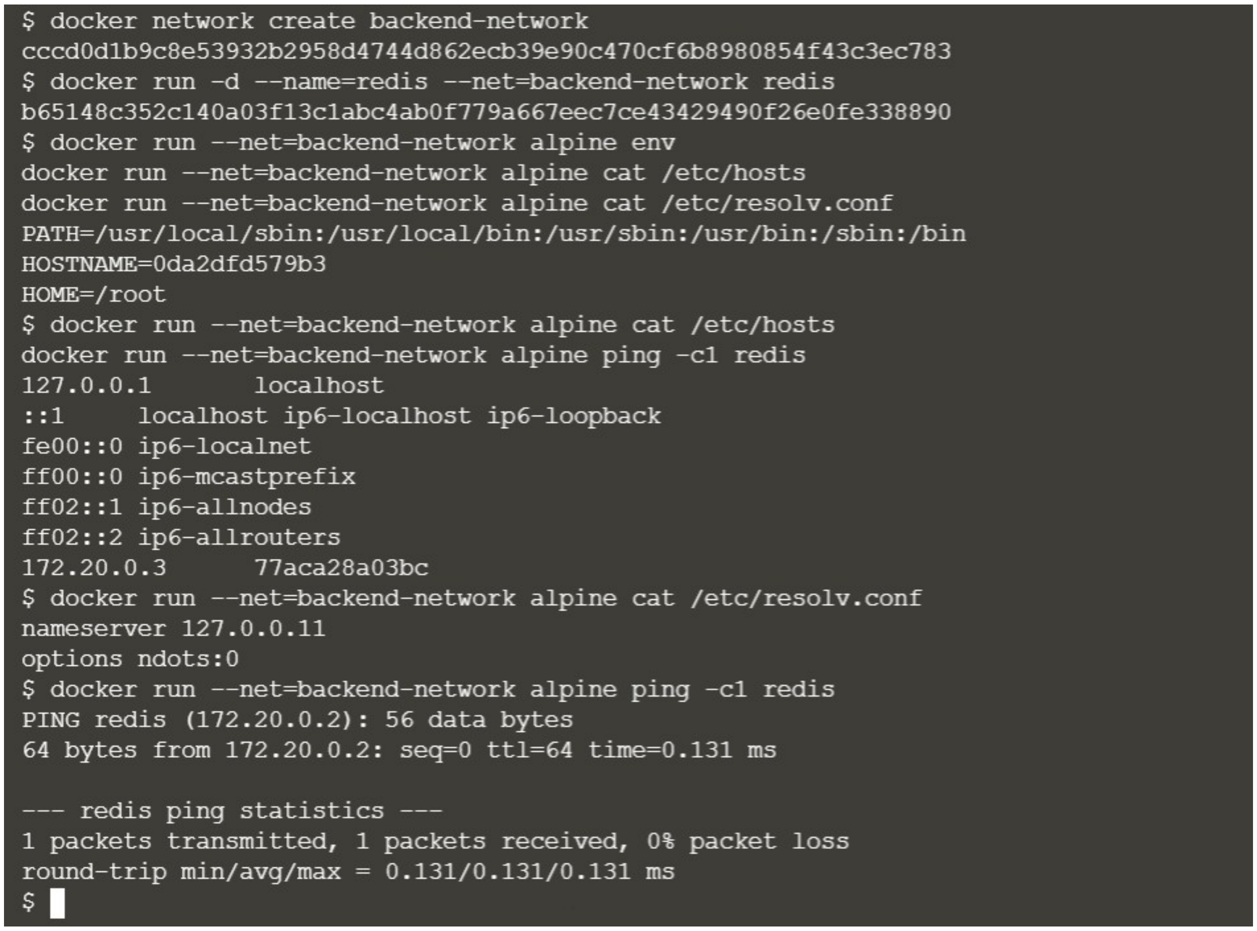
```
            "a18066204754"
        ],
        "NetworkID": "b7b43950036c40b0f9bec252d7c9ad79ef1
        "EndpointID": "0bda81d19a819b43ae54b1d2552f9ee983
        "Gateway": "172.19.0.1",
        "IPAddress": "172.19.0.2",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "MacAddress": "02:42:ac:13:00:02",
        "DriverOpts": null
    }
```



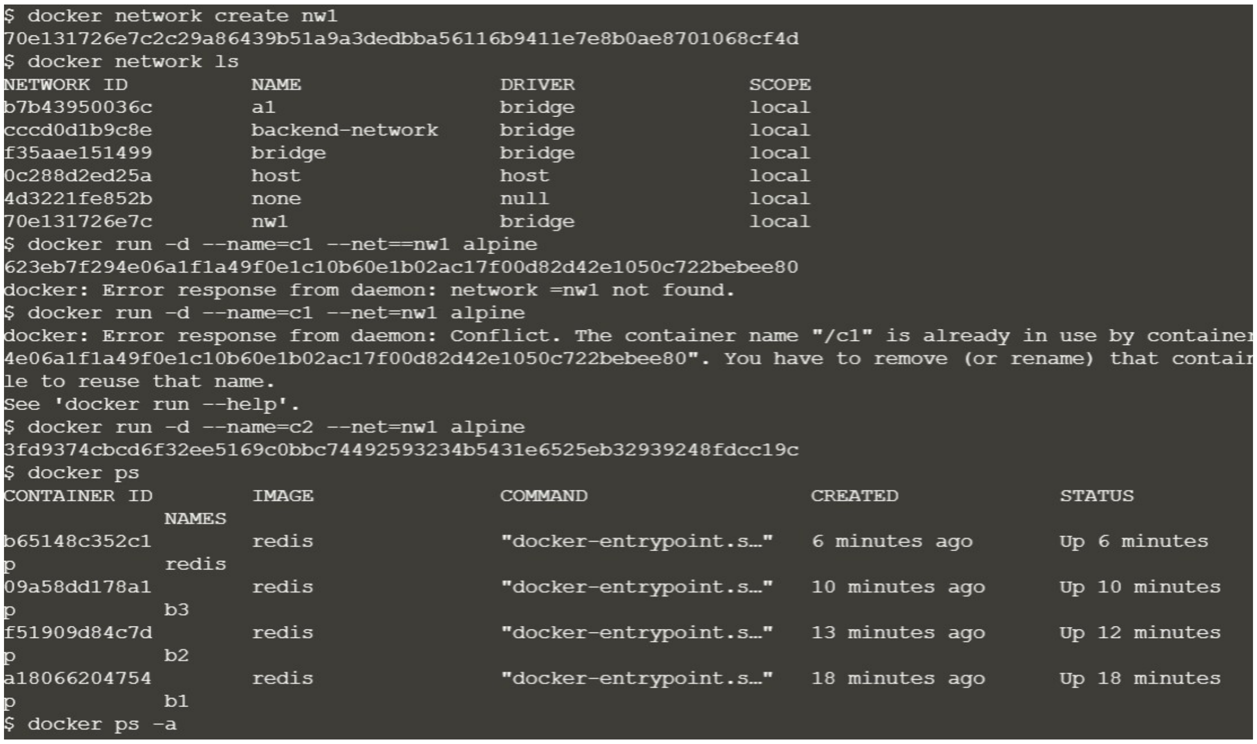
```
$ docker run -d --name=b2 --net=a1 redis
f51909d84c7d23264225be8bb8293c58914abef5c81da3f730500832b0e7130a
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
               NAMES
f51909d84c7d        redis               "docker-entrypoint.s…"   6 seconds ago       Up 4 secon
p              b2
a18066204754        redis               "docker-entrypoint.s…"   5 minutes ago       Up 5 minut
p              b1
$ docker inspect f5
[
    {
        "Id": "f51909d84c7d23264225be8bb8293c58914abef5c81da3f730500832b0e7130a",
        "Created": "2021-02-05T06:18:43.192236106Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "redis-server"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 1923,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2021-02-05T06:18:43.692506156Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
```



```
    "f51909d84c7d"
],
"NetworkID": "b7b43950036c40b0f9bec252d7c9ad79ef163f2474e
"EndpointID": "e946ac1b0df4a58863585a1c65493cea63a6d6b1b9
"Gateway": "172.19.0.1",
"IPAddress": "172.19.0.3",
"IPPrefixLen": 16,
"IPv6Gateway": "",
"GlobalIPv6Address": "",
"GlobalIPv6PrefixLen": 0,
"MacAddress": "02:42:ac:13:00:03",
"DriverOpts": null
```



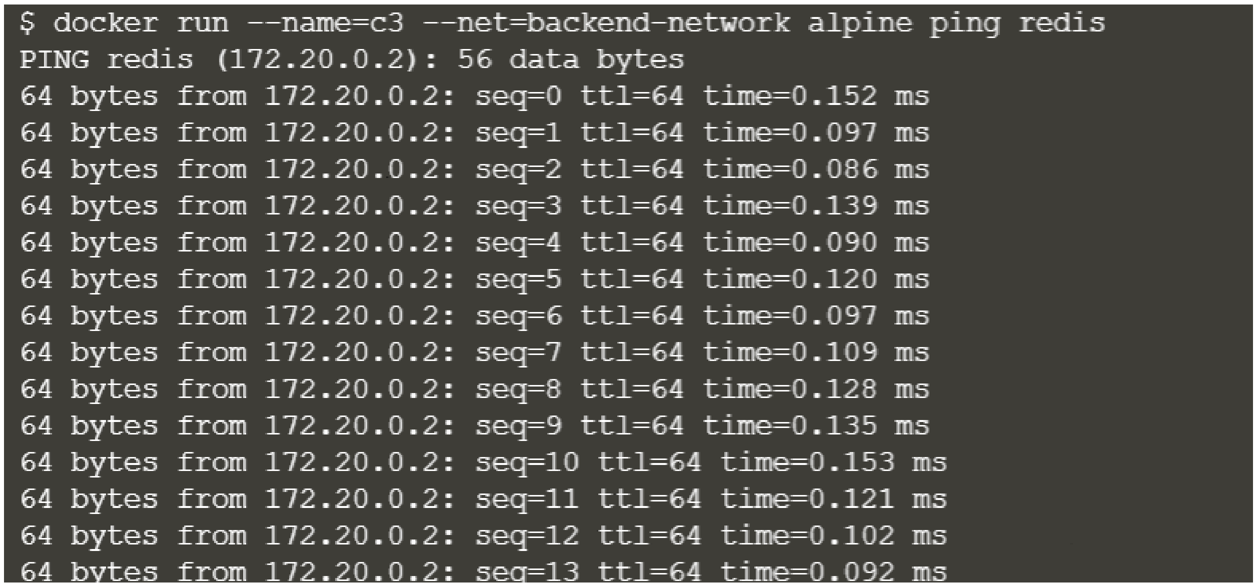
```
$ docker network create backend-network
cccd0d1b9c8e53932b2958d4744d862ecb39e90c470cf6b8980854f43c3ec783
$ docker run -d --name=redis --net=backend-network redis
b65148c352c140a03f13c1abc4ab0f779a667eec7ce43429490f26e0fe338890
$ docker run --net=backend-network alpine env
docker run --net=backend-network alpine cat /etc/hosts
docker run --net=backend-network alpine cat /etc/resolv.conf
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=0da2dfd579b3
HOME=/root
$ docker run --net=backend-network alpine cat /etc/hosts
docker run --net=backend-network alpine ping -c1 redis
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.20.0.3      77aca28a03bc
$ docker run --net=backend-network alpine cat /etc/resolv.conf
nameserver 127.0.0.11
options ndots:0
$ docker run --net=backend-network alpine ping -c1 redis
PING redis (172.20.0.2): 56 data bytes
64 bytes from 172.20.0.2: seq=0 ttl=64 time=0.131 ms

--- redis ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.131/0.131/0.131 ms
$ █
```



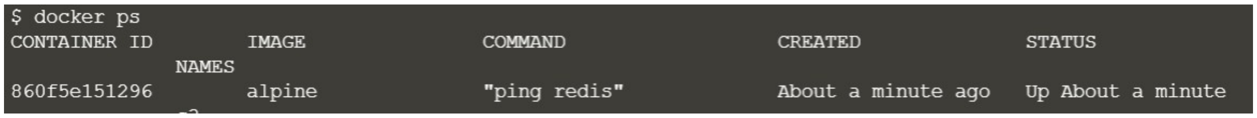
```
$ docker network create nw1
70e131726e7c2c29a86439b51a9a3dedbba56116b9411e7e8b0ae8701068cf4d
$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
b7b43950036c        a1                  bridge              local
cccd0d1b9c8e        backend-network     bridge              local
f35aae151499        bridge              bridge              local
0c288d2ed25a        host                host                local
4d3221fe852b        none                null                local
70e131726e7c        nw1                 bridge              local
$ docker run -d --name=c1 --net==nw1 alpine
623eb7f294e06a1f1a49f0e1c10b60e1b02ac17f00d82d42e1050c722bebee80
docker: Error response from daemon: network =nw1 not found.
$ docker run -d --name=c1 --net=nw1 alpine
docker: Error response from daemon: Conflict. The container name "/c1" is already in use by containe
4e06a1f1a49f0e1c10b60e1b02ac17f00d82d42e1050c722bebee80". You have to remove (or rename) that contai
le to reuse that name.
See 'docker run --help'.
$ docker run -d --name=c2 --net=nw1 alpine
3fd9374cbcd6f32ee5169c0bbc74492593234b5431e6525eb32939248fdcc19c
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
             NAMES
b65148c352c1        redis               "docker-entrypoint.s…"   6 minutes ago       Up 6 minutes
p            redis
09a58dd178a1        redis               "docker-entrypoint.s…"   10 minutes ago      Up 10 minutes
p            b3
f51909d84c7d        redis               "docker-entrypoint.s…"   13 minutes ago      Up 12 minutes
p            b2
a18066204754        redis               "docker-entrypoint.s…"   18 minutes ago      Up 18 minutes
p            b1
$ docker ps -a
```

Ping network

```
$ docker run –name=c3 –net=backend-network alpine ping redis
```



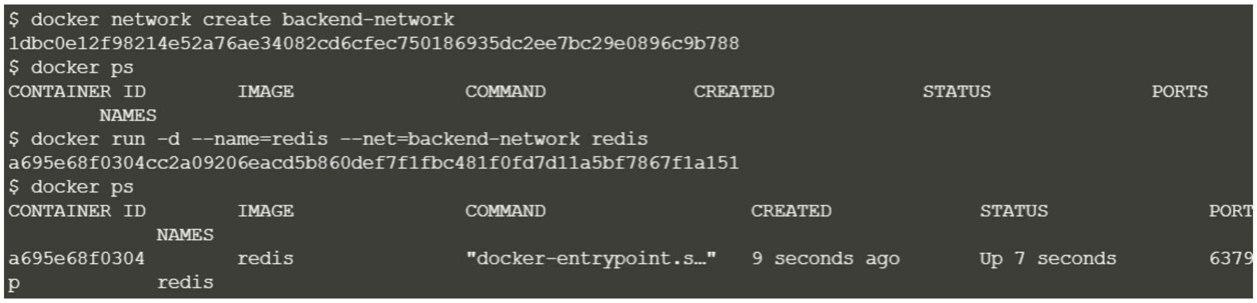
```
$ docker run --name=c3 --net=backend-network alpine ping redis
PING redis (172.20.0.2): 56 data bytes
64 bytes from 172.20.0.2: seq=0 ttl=64 time=0.152 ms
64 bytes from 172.20.0.2: seq=1 ttl=64 time=0.097 ms
64 bytes from 172.20.0.2: seq=2 ttl=64 time=0.086 ms
64 bytes from 172.20.0.2: seq=3 ttl=64 time=0.139 ms
64 bytes from 172.20.0.2: seq=4 ttl=64 time=0.090 ms
64 bytes from 172.20.0.2: seq=5 ttl=64 time=0.120 ms
64 bytes from 172.20.0.2: seq=6 ttl=64 time=0.097 ms
64 bytes from 172.20.0.2: seq=7 ttl=64 time=0.109 ms
64 bytes from 172.20.0.2: seq=8 ttl=64 time=0.128 ms
64 bytes from 172.20.0.2: seq=9 ttl=64 time=0.135 ms
64 bytes from 172.20.0.2: seq=10 ttl=64 time=0.153 ms
64 bytes from 172.20.0.2: seq=11 ttl=64 time=0.121 ms
64 bytes from 172.20.0.2: seq=12 ttl=64 time=0.102 ms
64 bytes from 172.20.0.2: seq=13 ttl=64 time=0.092 ms
```



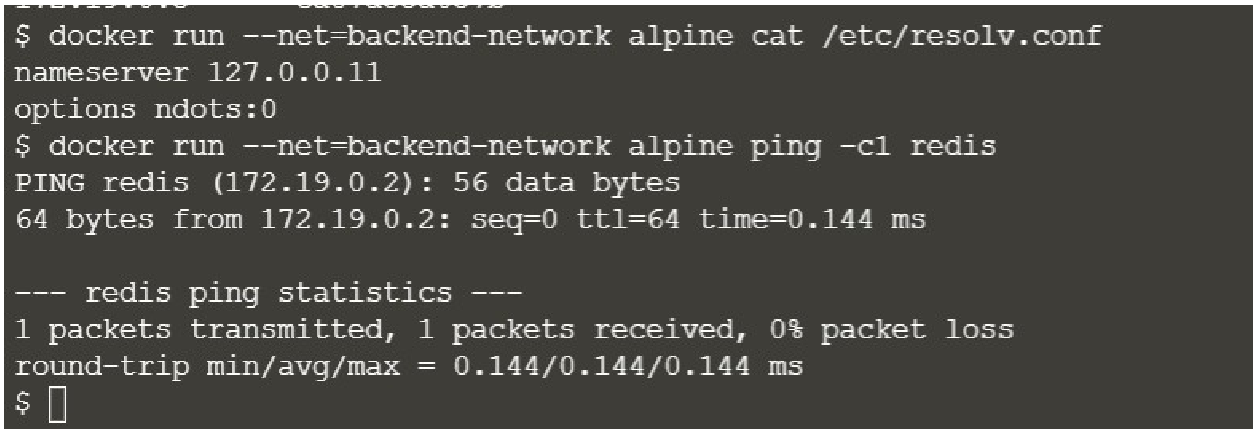
```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED              STATUS
        NAMES
860f5e151296        alpine              "ping redis"        About a minute ago   Up About a minute
```

Create a network

```
$ docker network create backend-network
```



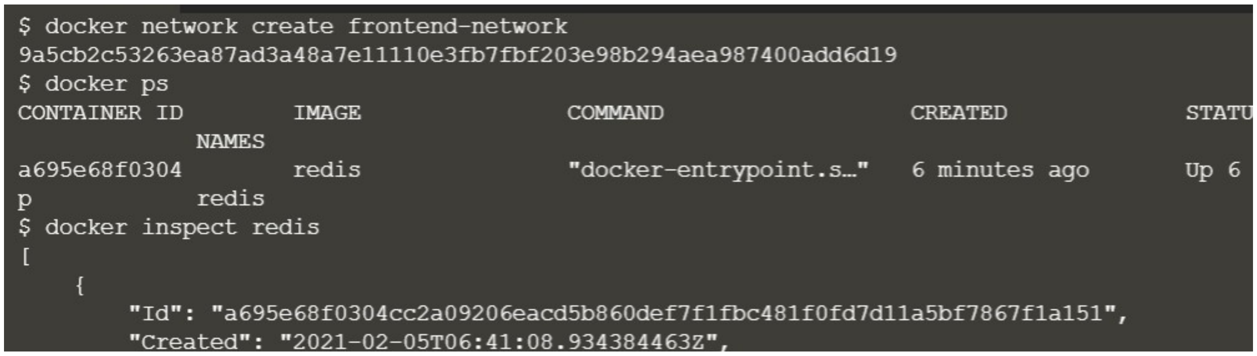
```
$ docker network create backend-network
1dbc0e12f98214e52a76ae34082cd6cfec750186935dc2ee7bc29e0896c9b788
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
        NAMES
$ docker run -d --name=redis --net=backend-network redis
a695e68f0304cc2a09206eacd5b860def7f1fbc481f0fd7d11a5bf7867f1a151
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORT
            NAMES
a695e68f0304        redis               "docker-entrypoint.s…"   9 seconds ago       Up 7 seconds        6379
p           redis
```



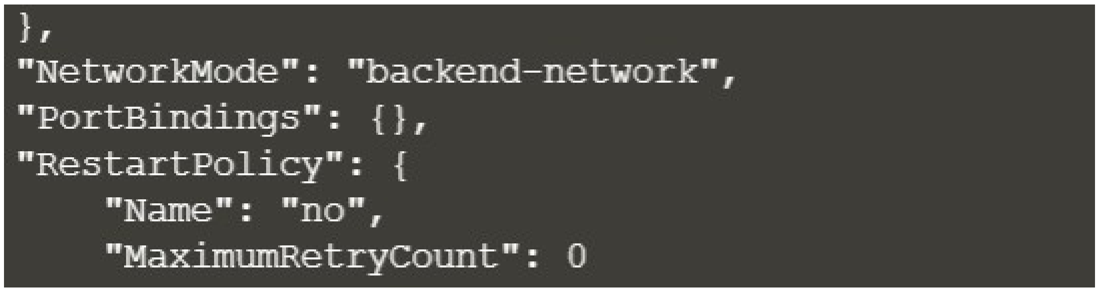
```
$ docker run --net=backend-network alpine cat /etc/resolv.conf
nameserver 127.0.0.11
options ndots:0
$ docker run --net=backend-network alpine ping -c1 redis
PING redis (172.19.0.2): 56 data bytes
64 bytes from 172.19.0.2: seq=0 ttl=64 time=0.144 ms

--- redis ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.144/0.144/0.144 ms
$ 
```

Create a network

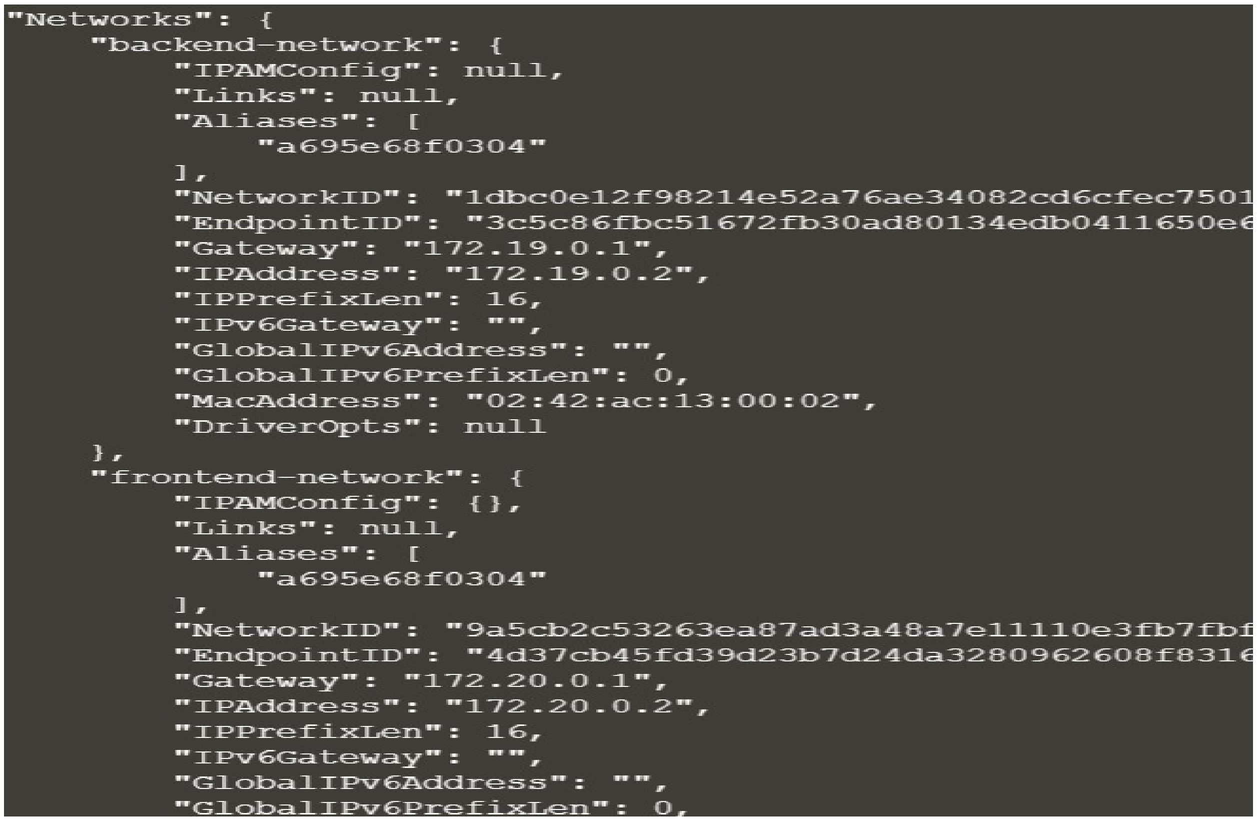
```
$ docker network create frontend-network
```



```
$ docker network create frontend-network
9a5cb2c53263ea87ad3a48a7e11110e3fb7fbf203e98b294aea987400add6d19
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATU
              NAMES
a695e68f0304        redis               "docker-entrypoint.s…"   6 minutes ago       Up 6
p             redis
$ docker inspect redis
[
    {
        "Id": "a695e68f0304cc2a09206eacd5b860def7f1fbc481f0fd7d11a5bf7867f1a151",
        "Created": "2021-02-05T06:41:08.934384463Z",
```

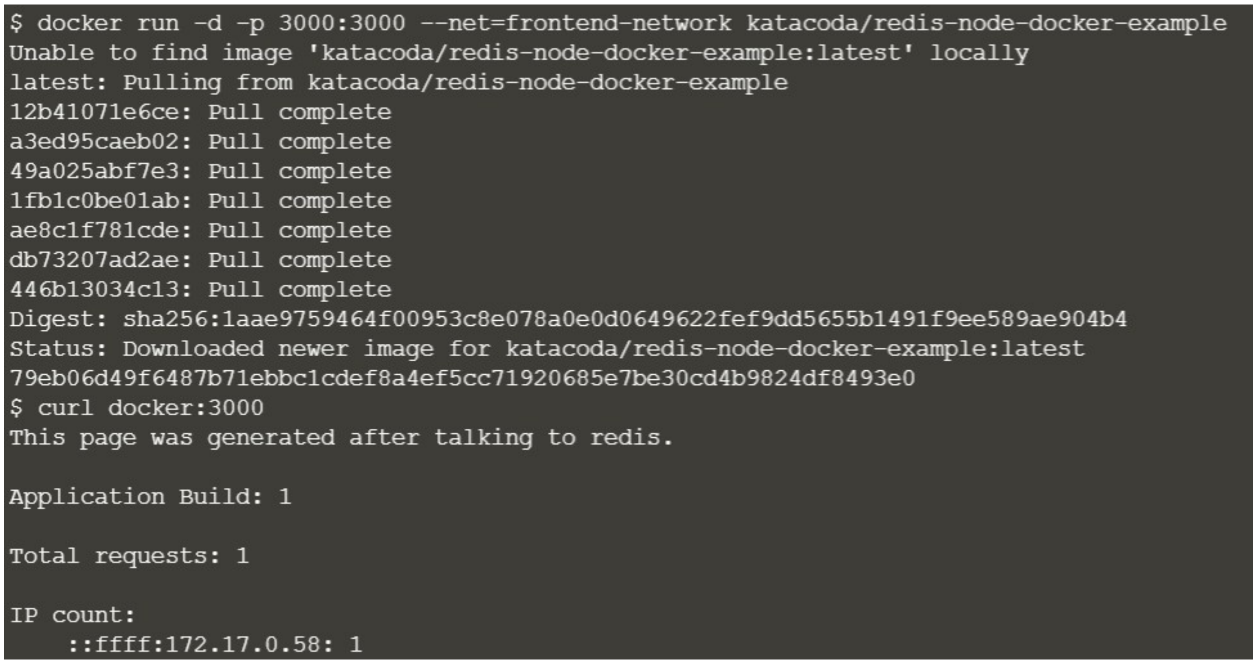


```
},
"NetworkMode": "backend-network",
"PortBindings": {},
"RestartPolicy": {
    "Name": "no",
    "MaximumRetryCount": 0
```



```
"Networks": {
    "backend-network": {
        "IPAMConfig": null,
        "Links": null,
        "Aliases": [
            "a695e68f0304"
        ],
        "NetworkID": "1dbc0e12f98214e52a76ae34082cd6cfec7501
        "EndpointID": "3c5c86fbc51672fb30ad80134edb0411650e6
        "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": [
            "a695e68f0304"
        ],
        "NetworkID": "9a5cb2c53263ea87ad3a48a7e11110e3fb7fbf
        "EndpointID": "4d37cb45fd39d23b7d24da3280962608f8316
        "Gateway": "172.20.0.1",
        "IPAddress": "172.20.0.2",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
```

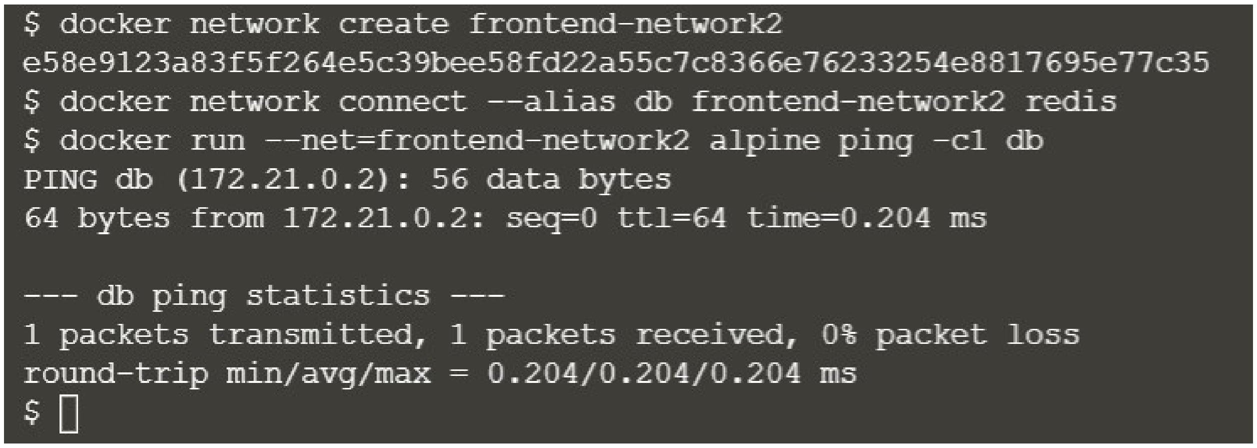
`$ docker run –d –p 3000:3000 –net=frontend-network katacoda/redis-node-docker-example`



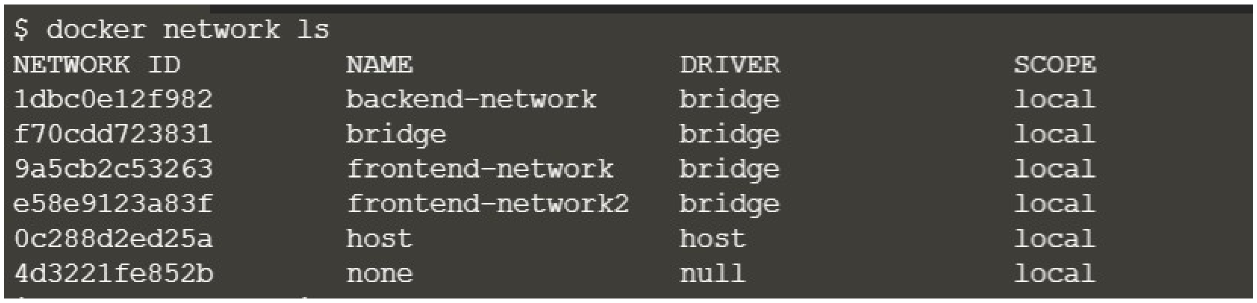
```
$ docker run -d -p 3000:3000 --net=frontend-network katacoda/redis-node-docker-example
Unable to find image 'katacoda/redis-node-docker-example:latest' locally
latest: Pulling from katacoda/redis-node-docker-example
12b41071e6ce: Pull complete
a3ed95caeb02: Pull complete
49a025abf7e3: Pull complete
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
79eb06d49f6487b71ebbc1cdef8a4ef5cc71920685e7be30cd4b9824df8493e0
$ curl docker:3000
This page was generated after talking to redis.

Application Build: 1

Total requests: 1

IP count:
    ::ffff:172.17.0.58: 1
```

To create a network

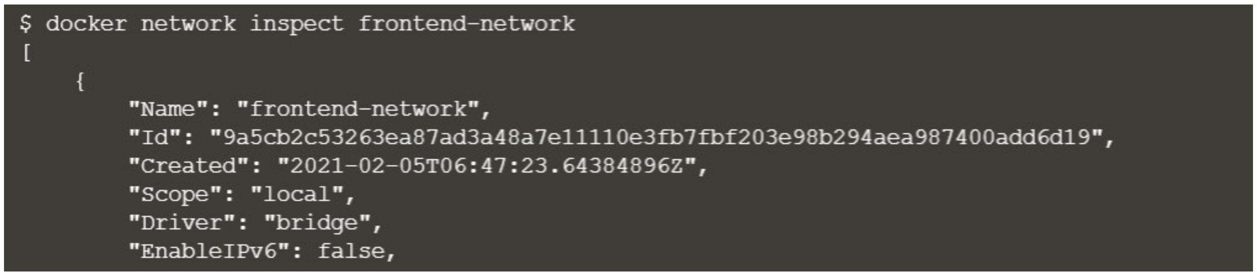
$ docker network create frontend-network



```
$ docker network create frontend-network2
e58e9123a83f5f264e5c39bee58fd22a55c7c8366e76233254e8817695e77c35
$ 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=0.204 ms

--- db ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.204/0.204/0.204 ms
$
```



```
$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
1dbc0e12f982        backend-network     bridge              local
f70cdd723831        bridge              bridge              local
9a5cb2c53263        frontend-network    bridge              local
e58e9123a83f        frontend-network2   bridge              local
0c288d2ed25a        host                host                local
4d3221fe852b        none                null                local
```

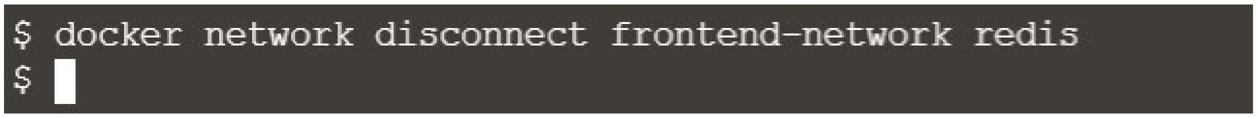


```
$ docker network inspect frontend-network
[
    {
        "Name": "frontend-network",
        "Id": "9a5cb2c53263ea87ad3a48a7e11110e3fb7fbf203e98b294aea987400add6d19",
        "Created": "2021-02-05T06:47:23.64384896Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
```

To disconnect from server

$ docker network disconnect frontend-network redis

(frontend-network is name of network)



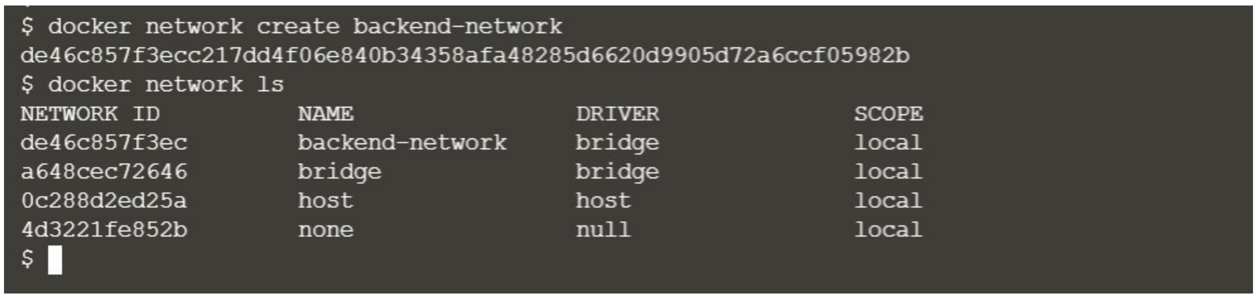
```
$ docker network disconnect frontend-network redis
$ 
```

Experiment 4

Aim: Create the network and connect with the container

Create Network:-

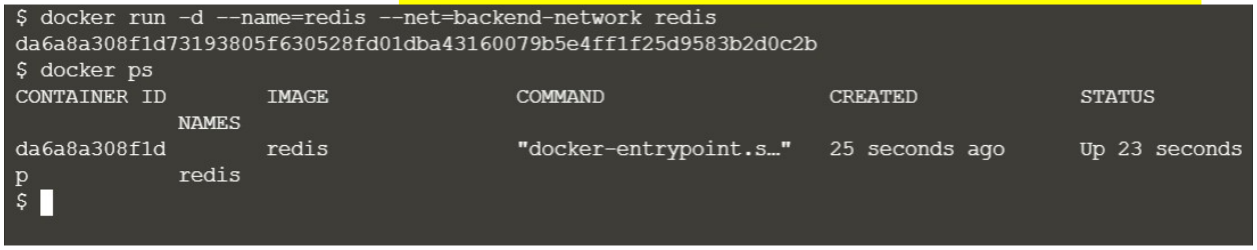
Command:- docker network create backend-network



```
$ docker network create backend-network
de46c857f3ecc217dd4f06e840b34358afa48285d6620d9905d72a6ccf05982b
$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
de46c857f3ec        backend-network     bridge              local
a648cec72646        bridge              bridge              local
0c288d2ed25a        host                host                local
4d3221fe852b        none                null                local
$
```

Connect To Network:-

Command:- docker run -d --name=redis --net=backend-network redis

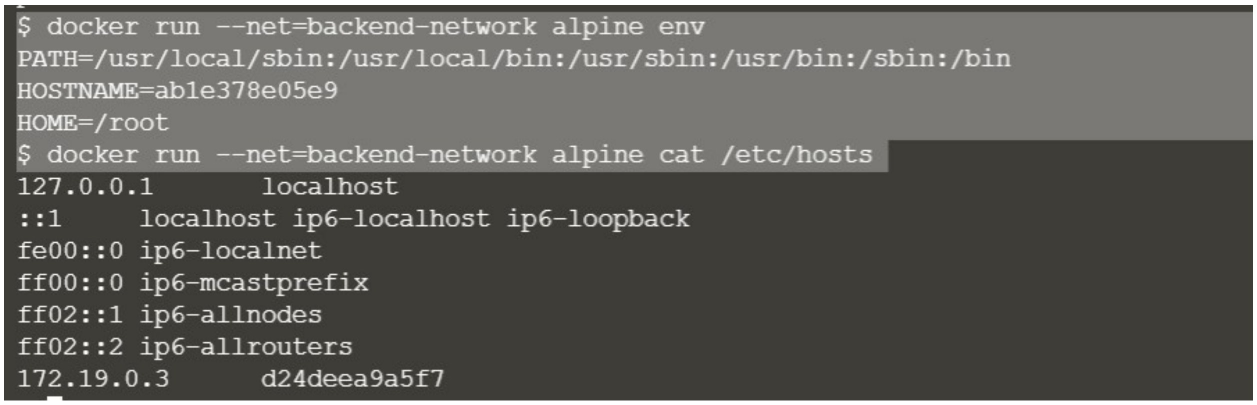


```
$ docker run -d --name=redis --net=backend-network redis
da6a8a308f1d73193805f630528fd01dba43160079b5e4ff1f25d9583b2d0c2b
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
             NAMES
da6a8a308f1d        redis               "docker-entrypoint.s…"   25 seconds ago      Up 23 seconds
p            redis
$
```

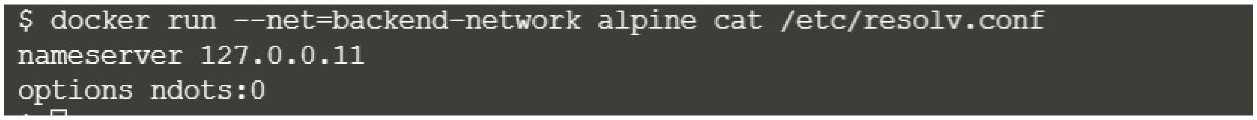
Explore:- The first thing you'll notice is that Docker no longer assigns environment variables or updates the hosts file of containers. Explore using the following two commands and you'll notice it no longer mentions other containers.

Command:- [1] $ docker run --net=backend-network alpine env

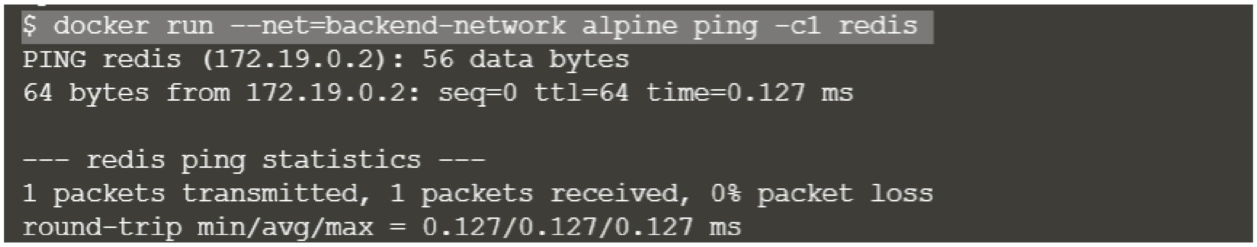
[2] $ docker run --net=backend-network alpine cat /etc/hosts


```
$ docker run --net=backend-network alpine env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=ab1e378e05e9
HOME=/root
$ docker run --net=backend-network alpine cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.19.0.3      d24deea9a5f7
```

The way containers can communicate via an Embedded DNS Server in Docker. This DNS server is assigned to all containers via the IP 127.0.0.11 and set in the *resolv.conf* file.

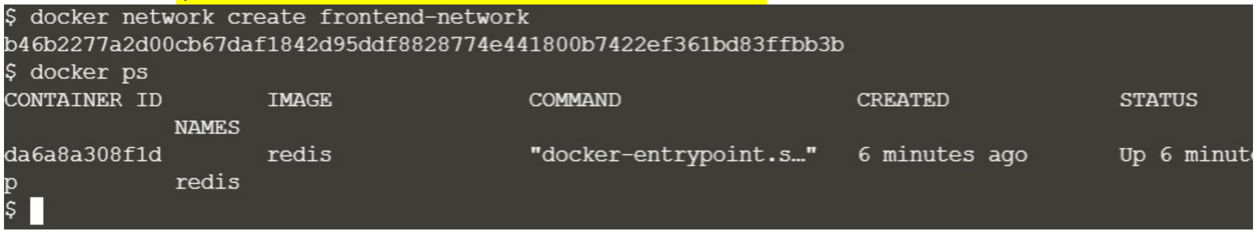

```
$ docker run --net=backend-network alpine cat /etc/resolv.conf
nameserver 127.0.0.11
options ndots:0
```

When containers access other container. The DNS server will return the IP of the container . In this case the name of redis will be *redis.backend-network*.


```
$ 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=0.127 ms

--- redis ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.127/0.127/0.127 ms
```

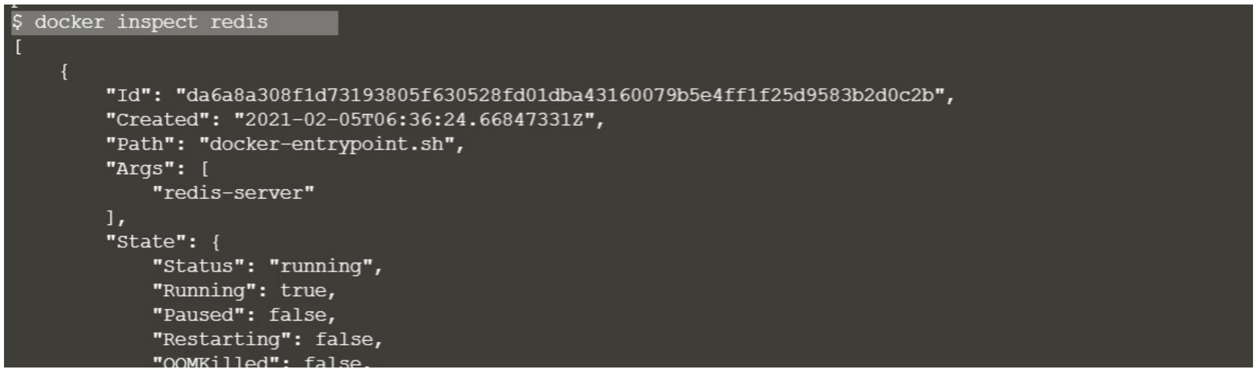
Again we create a new network with the help of this command which name is *frontend-network.*

Command- $ docker network create frontend-network

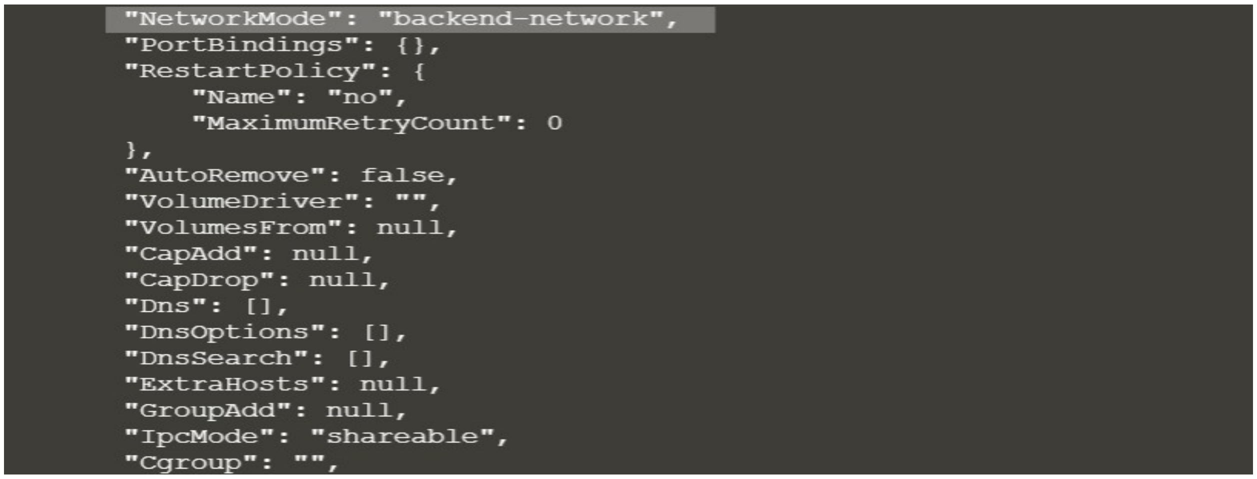

```
$ docker network create frontend-network
b46b2277a2d00cb67daf1842d95ddf8828774e441800b7422ef361bd83ffbb3b
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
           NAMES
da6a8a308f1d        redis               "docker-entrypoint.s…"   6 minutes ago       Up 6 minut
p          redis
$ 
```

- Command:- $docker inspect “container name”

Docker inspect provides detailed information on constructs controlled by Docker.



```
$ docker inspect redis
[
    {
        "Id": "da6a8a308f1d73193805f630528fd01dba43160079b5e4ff1f25d9583b2d0c2b",
        "Created": "2021-02-05T06:36:24.66847331Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "redis-server"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
```



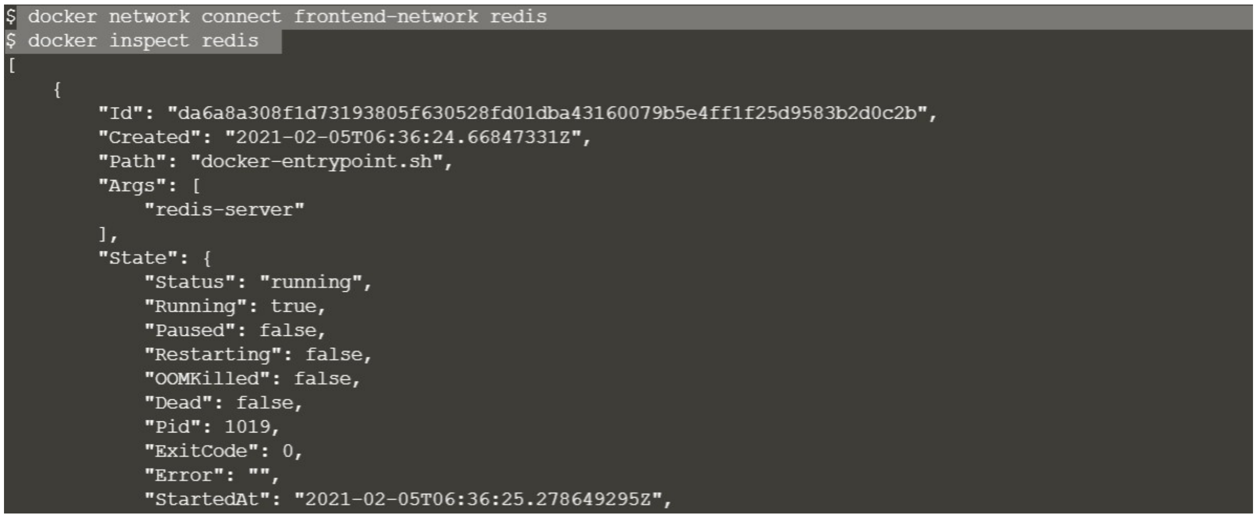
```
            "NetworkMode": "backend-network",
            "PortBindings": {},
            "RestartPolicy": {
                "Name": "no",
                "MaximumRetryCount": 0
            },
            "AutoRemove": false,
            "VolumeDriver": "",
            "VolumesFrom": null,
            "CapAdd": null,
            "CapDrop": null,
            "Dns": [],
            "DnsOptions": [],
            "DnsSearch": [],
            "ExtraHosts": null,
            "GroupAdd": null,
            "IpcMode": "shareable",
            "Cgroup": "",
```

Command:- $ docker network connect frontend-network redis

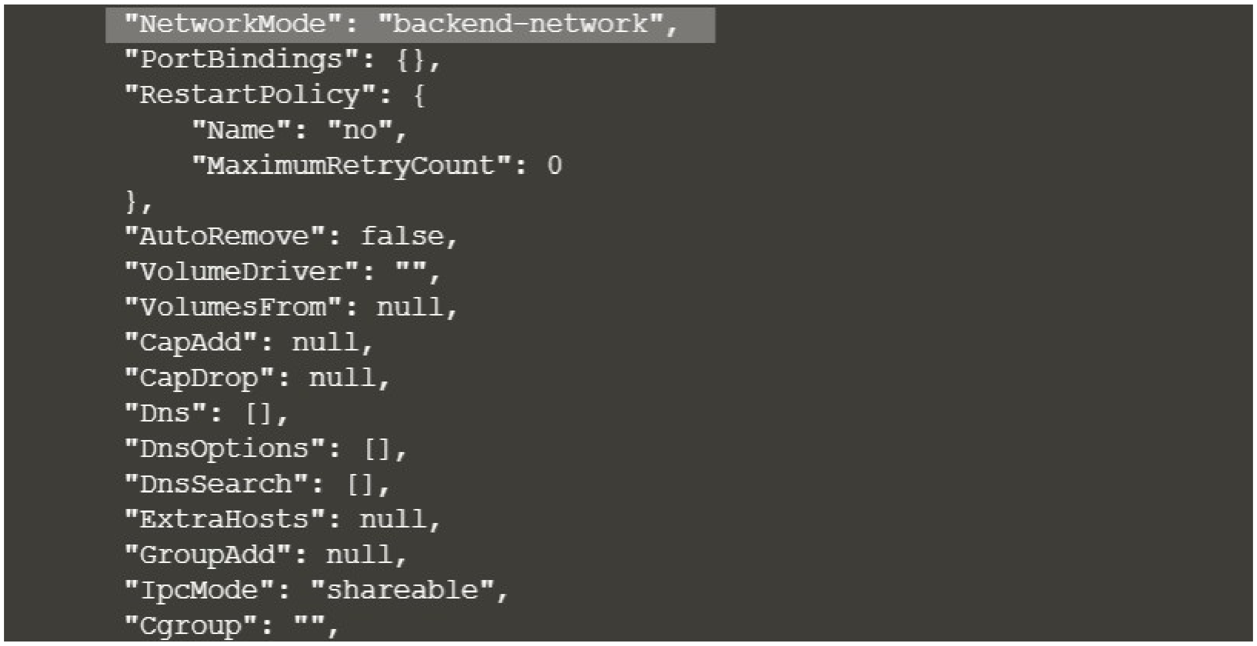
$ docker inspect redis

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

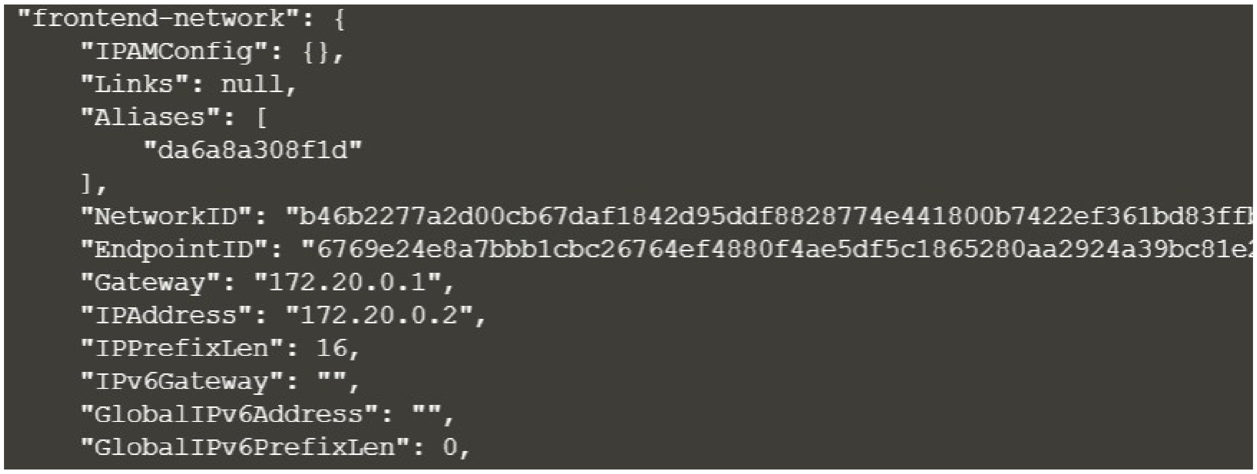
- Now we can see the backend and frontend network.



```
$ docker network connect frontend-network redis
$ docker inspect redis
[
    {
        "Id": "da6a8a308f1d73193805f630528fd01dba43160079b5e4ff1f25d9583b2d0c2b",
        "Created": "2021-02-05T06:36:24.66847331Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "redis-server"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 1019,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2021-02-05T06:36:25.278649295Z",
```



```
            "NetworkMode": "backend-network",
            "PortBindings": {},
            "RestartPolicy": {
                "Name": "no",
                "MaximumRetryCount": 0
            },
            "AutoRemove": false,
            "VolumeDriver": "",
            "VolumesFrom": null,
            "CapAdd": null,
            "CapDrop": null,
            "Dns": [],
            "DnsOptions": [],
            "DnsSearch": [],
            "ExtraHosts": null,
            "GroupAdd": null,
            "IpcMode": "shareable",
            "Cgroup": "",
```

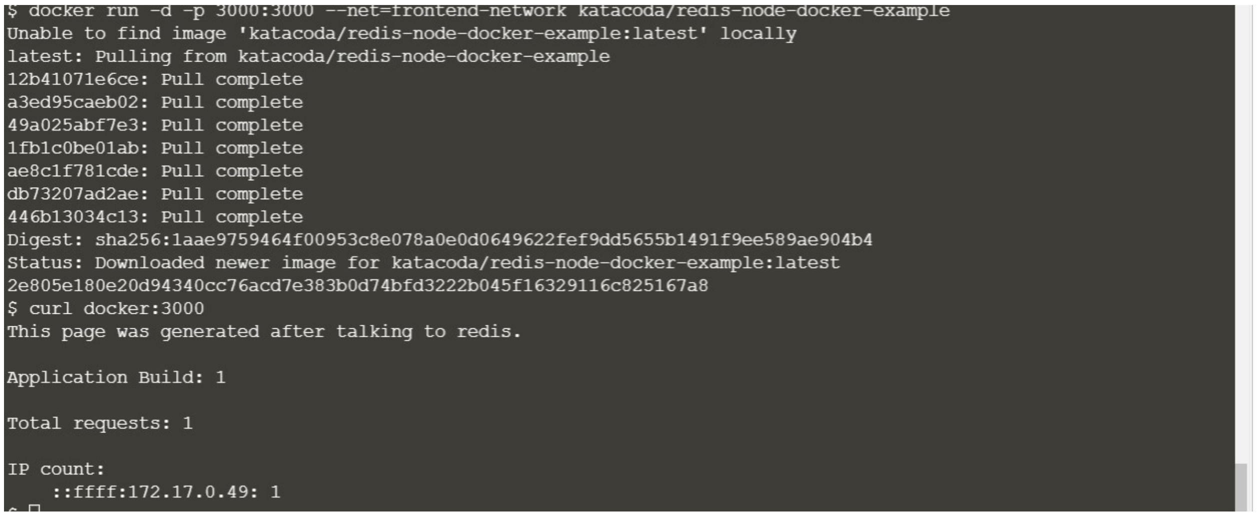


```
"frontend-network": {
    "IPAMConfig": {},
    "Links": null,
    "Aliases": [
        "da6a8a308f1d"
    ],
    "NetworkID": "b46b2277a2d00cb67daf1842d95ddf8828774e441800b7422ef361bd83ff
    "EndpointID": "6769e24e8a7bbb1cbc26764ef4880f4ae5df5c1865280aa2924a39bc81e
    "Gateway": "172.20.0.1",
    "IPAddress": "172.20.0.2",
    "IPPrefixLen": 16,
    "IPv6Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
```

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

$ curl docker:3000



```
$ 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
2e805e180e20d94340cc76acd7e383b0d74bfd3222b045f16329116c825167a8
$ curl docker:3000
This page was generated after talking to redis.

Application Build: 1

Total requests: 1

IP count:
    ::ffff:172.17.0.49: 1
```

Create Aliases:-

_ The other approach is to provide an alias when connecting a container to a network.

Connect container with Alias:-

Create a new network *frontend-network2*_

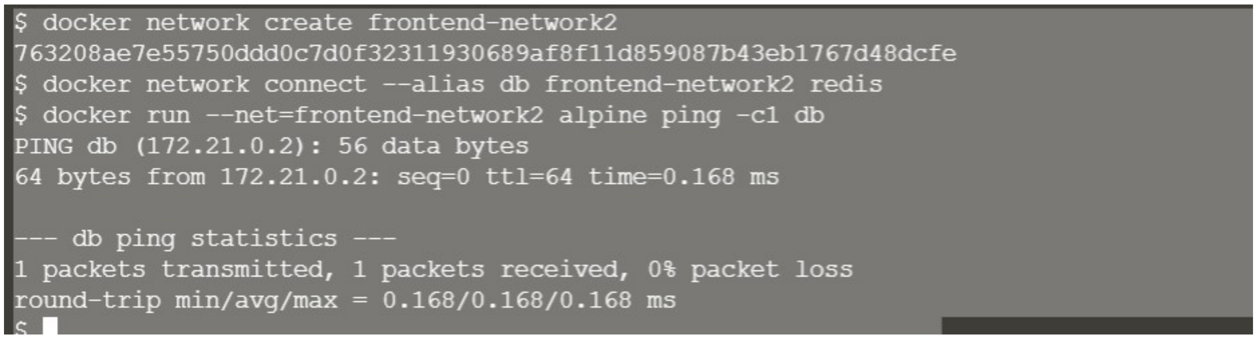
Command:- $ docker network create frontend-network2

Now The following command will connect our Redis instance to the frontend-network with the alias of *db*.

Command:- $ docker network connect --alias db frontend-network2 redis

When containers attempt to access a service via the name db, they will be given the IP address of our Redis container.

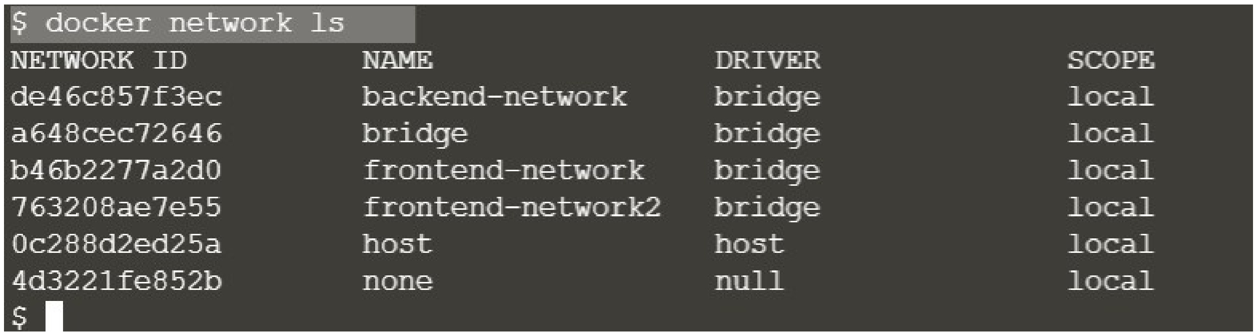
$ docker run --net=frontend-network2 alpine ping -c1 db



```
$ docker network create frontend-network2
763208ae7e55750ddd0c7d0f32311930689af8f11d859087b43eb1767d48dcfe
$ 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=0.168 ms

--- db ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.168/0.168/0.168 ms
$
```

To show the list of all the created network:-

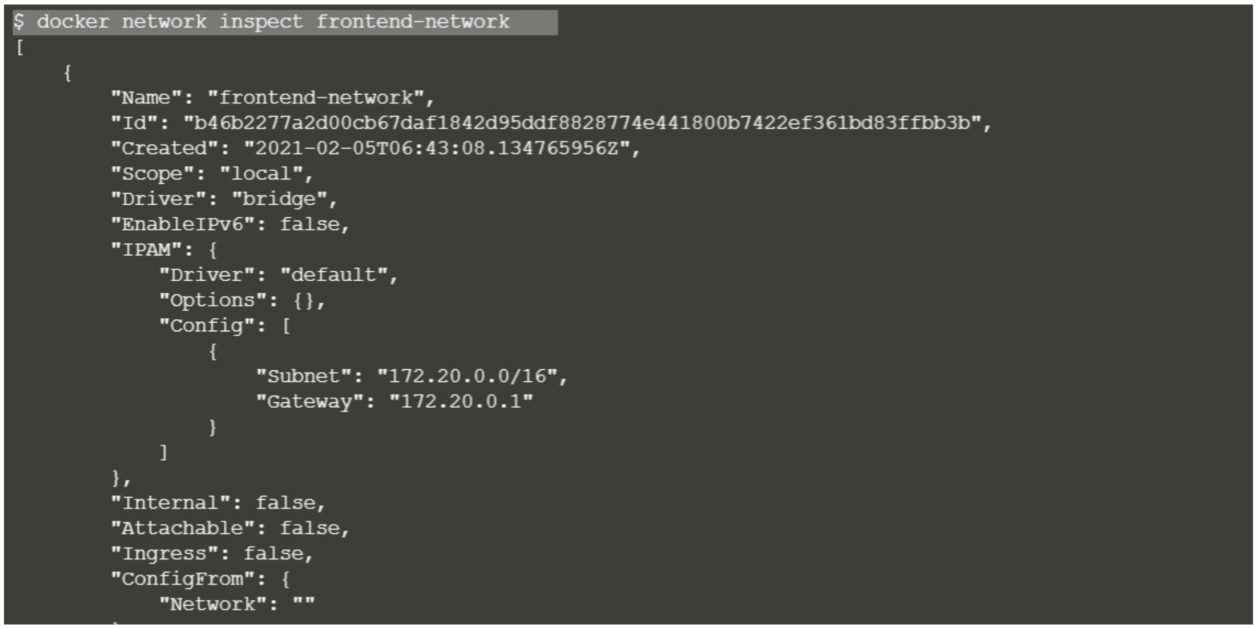
Command:- $ docker network ls



```
$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
de46c857f3ec        backend-network     bridge              local
a648cec72646        bridge              bridge              local
b46b2277a2d0        frontend-network    bridge              local
763208ae7e55        frontend-network2   bridge              local
0c288d2ed25a        host                host                local
4d3221fe852b        none                null                local
$
```

We can see all the information about frontend-network with the help of this command.

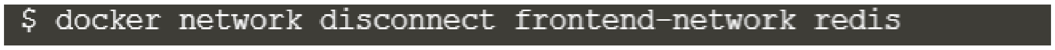
Command:- $ docker network inspect frontend-network



```
$ docker network inspect frontend-network
[
    {
        "Name": "frontend-network",
        "Id": "b46b2277a2d00cb67daf1842d95ddf8828774e441800b7422ef361bd83ffbb3b",
        "Created": "2021-02-05T06:43:08.134765956Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.20.0.0/16",
                    "Gateway": "172.20.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
```

__Disconnect the container:-__

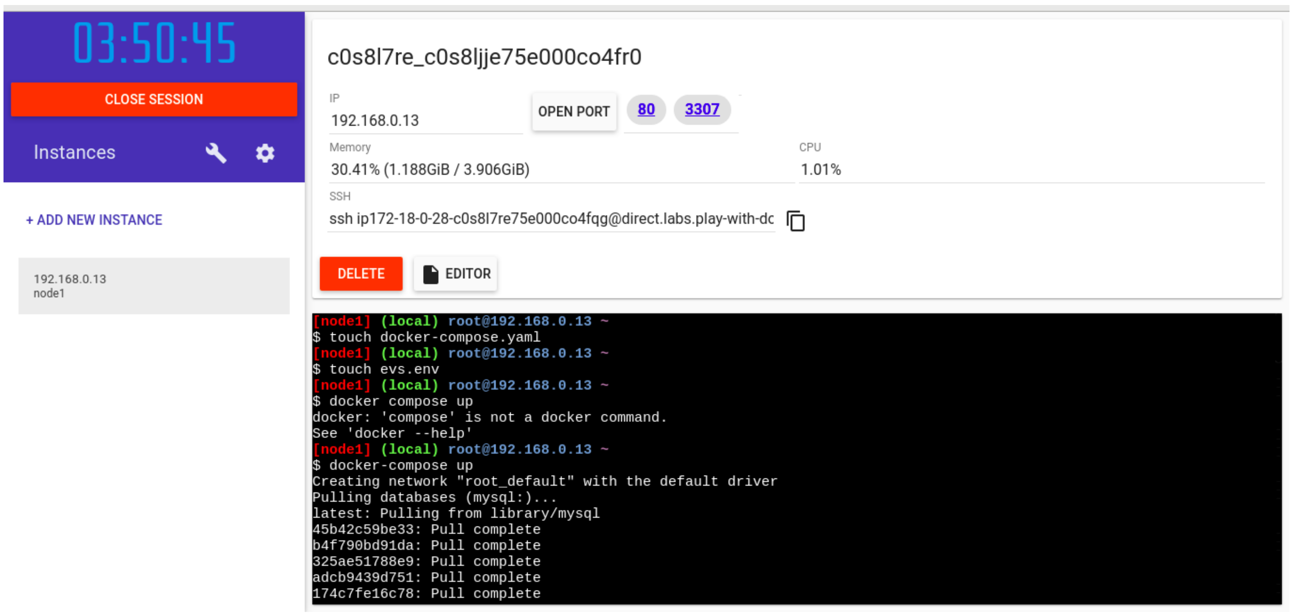
The following command disconnects the redis container from the *frontend-network*



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

Experiment 5

Aim: Create and run multi container application using docker compose



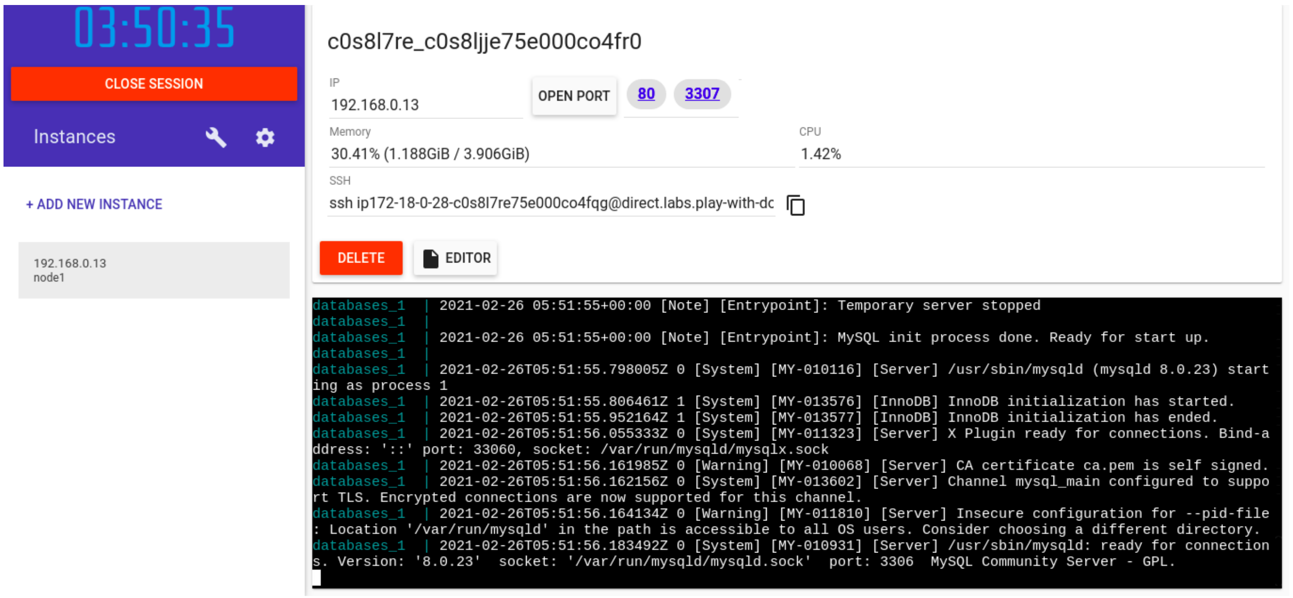
192.168.0.13

30.41% (1.188GiB / 3.906GiB)

1.01%

ssh ip172-18-0-28-c0s8l7re75e000co4fqg@direct.labs.play-with-dc

```
[node1] (local) root@192.168.0.13 ~
$ touch docker-compose.yaml
[node1] (local) root@192.168.0.13 ~
$ touch evs.env
[node1] (local) root@192.168.0.13 ~
$ docker compose up
docker: 'compose' is not a docker command.
See 'docker --help'
[node1] (local) root@192.168.0.13 ~
$ docker-compose up
Creating network "root_default" with the default driver
Pulling databases (mysql:)...
latest: Pulling from library/mysql
45b42c59be33: Pull complete
b4f790bd91da: Pull complete
325ae51788e9: Pull complete
adcb9439d751: Pull complete
174c7fe16c78: Pull complete
```



192.168.0.13

30.41% (1.188GiB / 3.906GiB)

1.42%

ssh ip172-18-0-28-c0s8l7re75e000co4fqg@direct.labs.play-with-dc

```
databases_1  | 2021-02-26 05:51:55+00:00 [Note] [Entrypoint]: Temporary server stopped
databases_1  |
databases_1  | 2021-02-26 05:51:55+00:00 [Note] [Entrypoint]: MySQL init process done. Ready for start up.
databases_1  |
databases_1  | 2021-02-26T05:51:55.798005Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.23) start
ing as process 1
databases_1  | 2021-02-26T05:51:55.806461Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
databases_1  | 2021-02-26T05:51:55.952164Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
databases_1  | 2021-02-26T05:51:56.055333Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-a
ddress: '::' port: 33060, socket: /var/run/mysqld/mysqlx.sock
databases_1  | 2021-02-26T05:51:56.161985Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
databases_1  | 2021-02-26T05:51:56.162156Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to suppo
rt TLS. Encrypted connections are now supported for this channel.
databases_1  | 2021-02-26T05:51:56.164134Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file
: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.
databases_1  | 2021-02-26T05:51:56.183492Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connection
s. Version: '8.0.23'  socket: '/var/run/mysqld/mysqld.sock'  port: 3306  MySQL Community Server - GPL.
```