LAPORAN PRAKTIKUM VIRTUALISASI KOMPUTER

DOCKER NETWORKS

Agus Pranata Marpaung

13323033

DIII TEKNOLOGI KOMPUTER

INSTITUT TEKNOLOGI DEL
FAKULTAS VOKASI

Judul Praktikum

Minggu/Sesi	:	XI/3
Kode Mata Kuliah	:	1032101
Nama Mata Kuliah	:	VIRTUALISASI KOMPUTER
Setoran	:	Jawaban dalam bentuk *softcopy*
Batas Waktu Setoran	:	*11 November 2024 jam 21:30*
Tujuan	:	1. Mahasiswa mampu membuat dan mengkonfigurasi jaringan antar container pada Docker

Petunjuk

Docker Networks

Teori

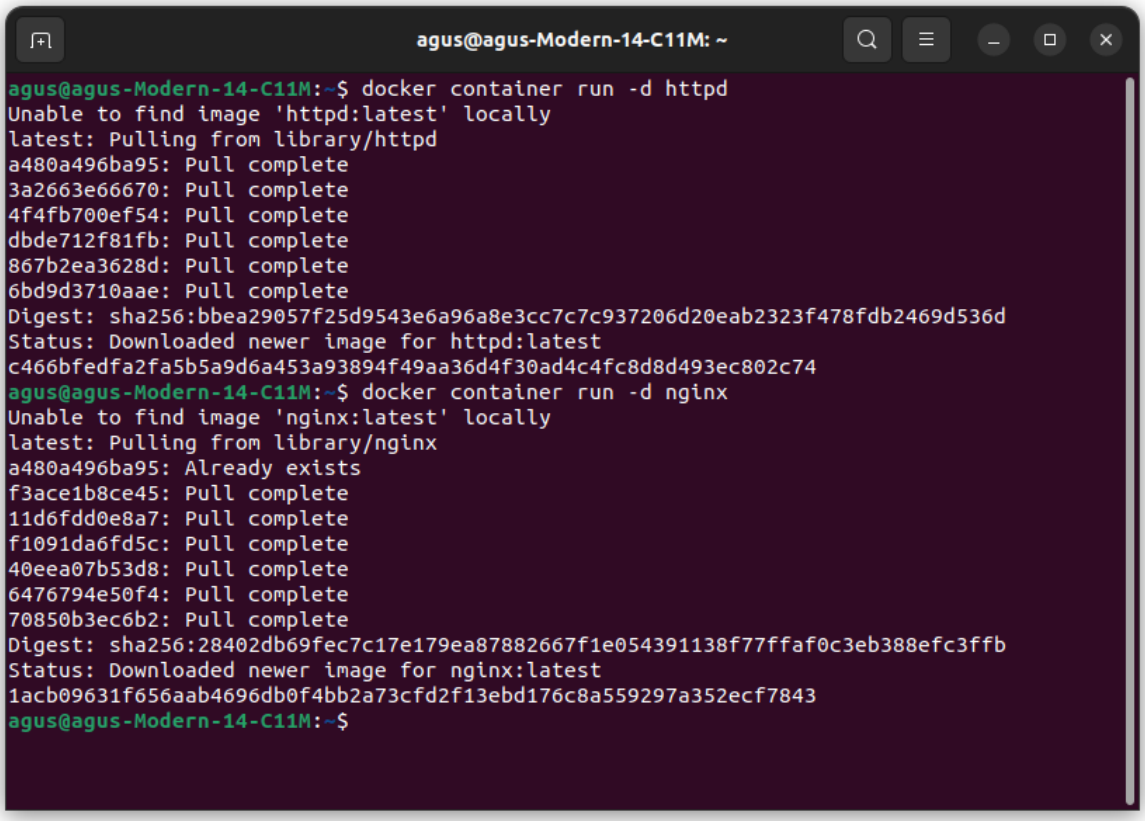
4 dari 6 Tipe Jaringan yang tersedia oleh Docker.

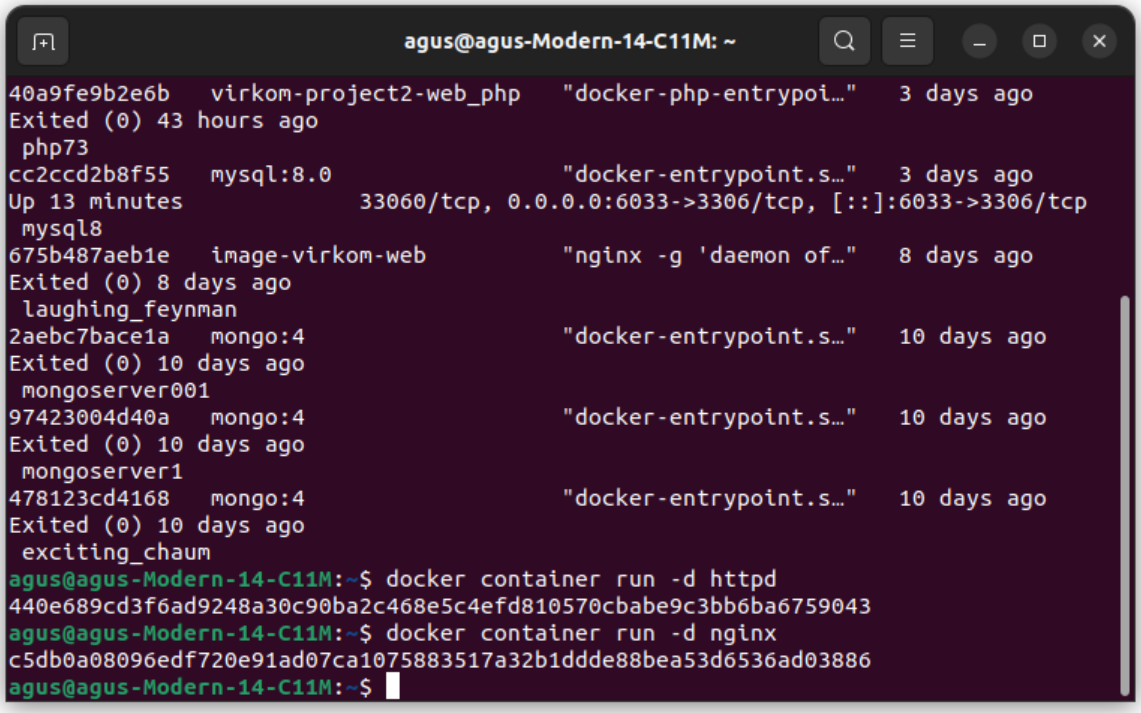
- **bridge**, jenis ini adalah jaringan default. Docker menghubungkan container ke jaringan ini secara default, kecuali memilih opsi driver yang berbeda saat menjalankan docker run -- network = <NETWORK.
- **none**, container tidak memiliki antarmuka jaringan.
- **host**, container yang berada di jaringan host. Tidak ada batas antara container dan host. Port yang ada yang dibuka di host dapat diakses langsung oleh container, dan sebaliknya.
- **macvlan**, jaringan ini memungkinkan pembuatan alamat MAC (alamat jaringan fisik).

Praktek

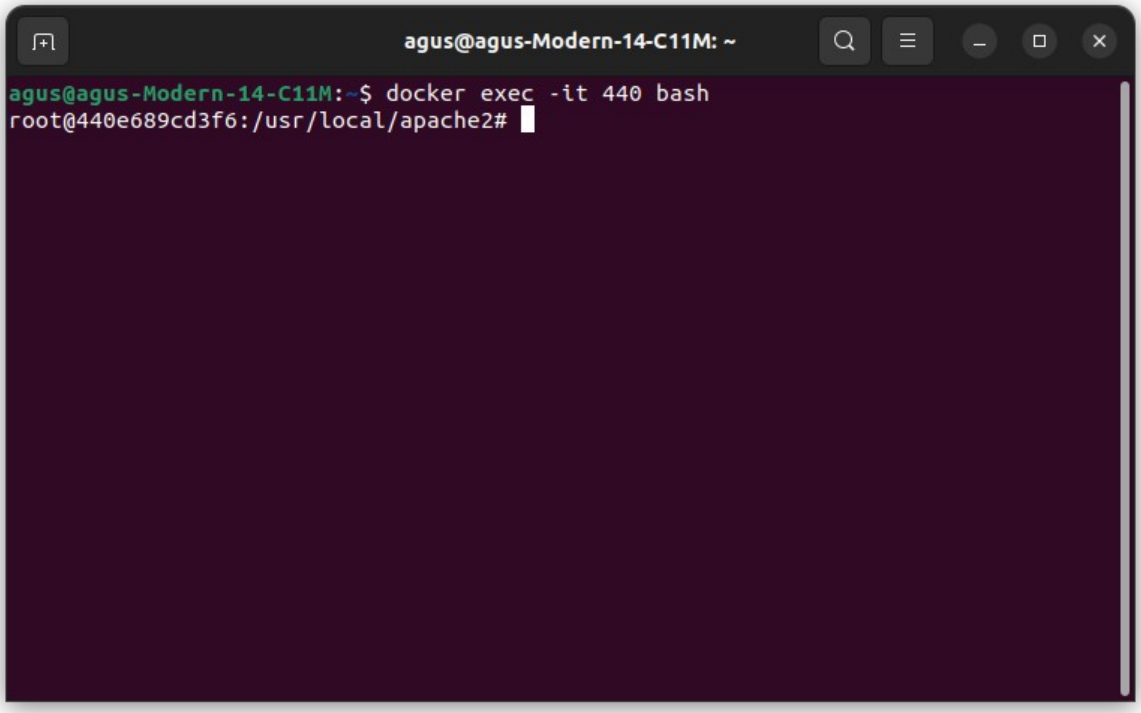
Dapatkah container berkomunikasi satu sama lain? Mari kita cari tahu.

1. Jalankan nginx dan httpd pada container

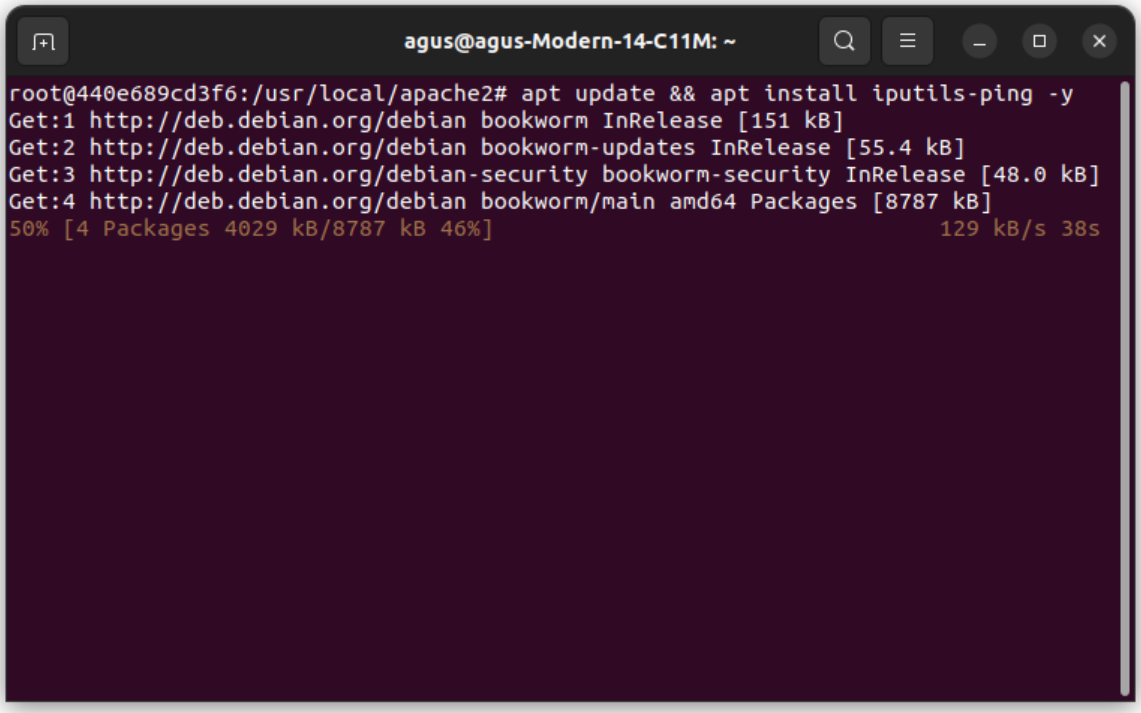




2. Mulaikan sesi bash didalam apache.

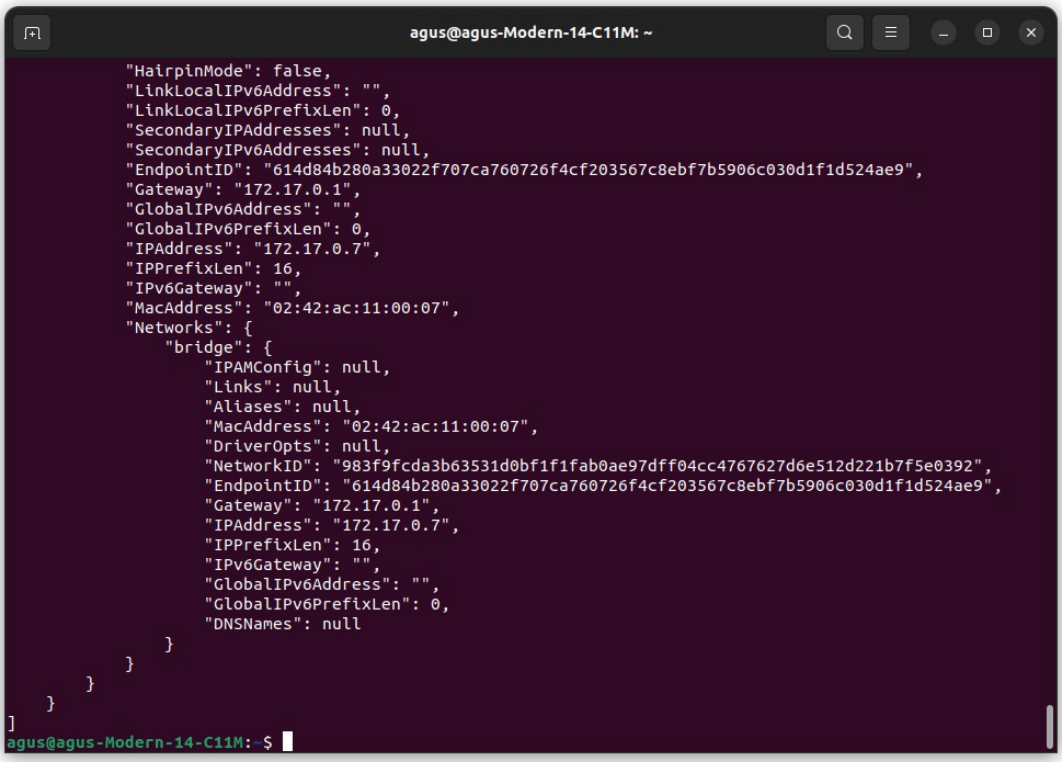


3. Install IPutils-ping untuk mencoba berkomunikasi dengan nginx container.



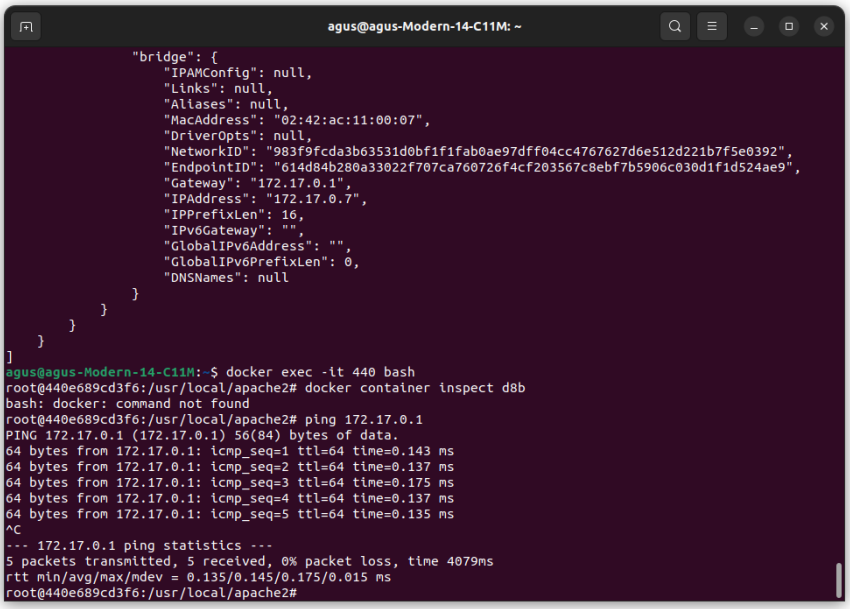
4. Mari kita cari tahu IP Address dari nginx container.

```
agus@agus-Modern-14-C11M:~$ docker container inspect d68
```



Catatan: docker inspect mengembalikan informasi terperinci tentang sumber daya yang kita tentukan dalam format JSON.

5. Ping nginx container menggunakan IP Address yang kita lihat sebelumnya.



Hal ini dimungkinkan karena adanya jaringan docker. Apa yang dimaksud dengan jaringan?

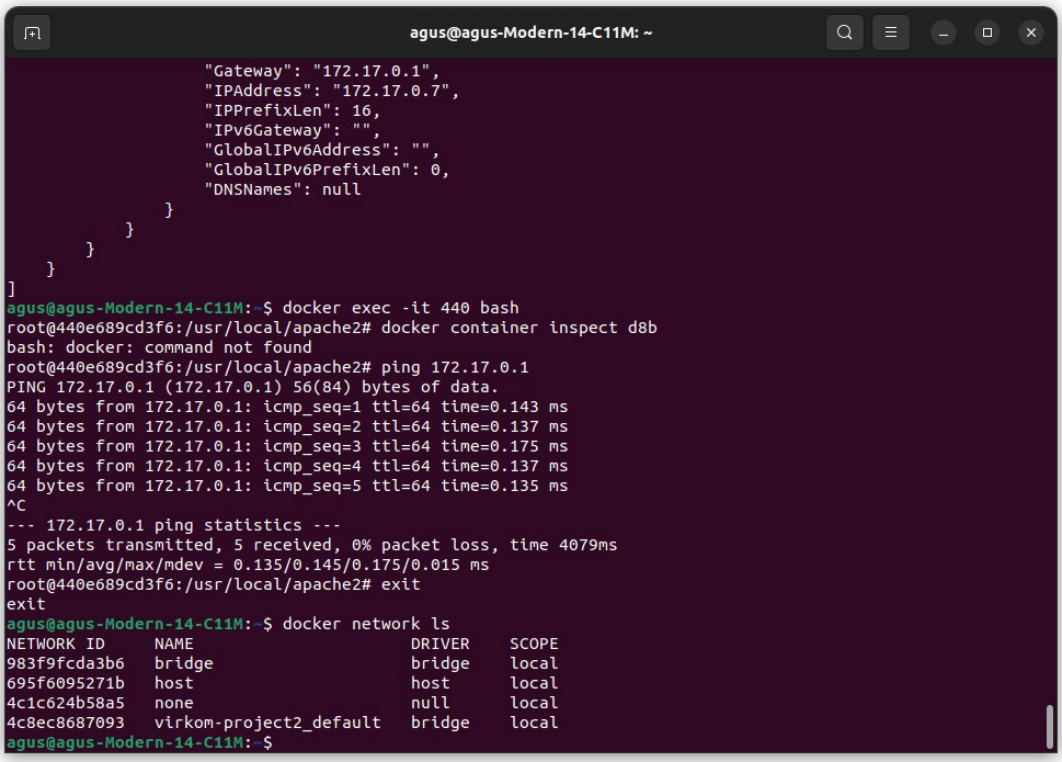
Sebuah jaringan dapat dianggap sebagai kumpulan container/simpul yang terhubung satu sama lain sedemikian rupa sehingga mereka dapat berbicara dengan container/simpul mana pun yang ada di dalam jaringan ini.

Tipe Jaringan

Docker memiliki 3 tipe:

- Bridge/docker0
- Host
- None

6. Mari kita lihat semua jaringan yang ada di docker.



Mari kita pahami satu per satu.

- **bridge/docker0**

  Anda pasti menyadari bahwa kami tidak perlu melakukan apa pun untuk membuat komunikasi menjadi mungkin. Ini karena :

  Secara default, setiap container menggunakan jaringan bridge kecuali ditentukan secara eksplisit.

  Ini adalah jaringan virtual pribadi yang bersifat internal untuk host sehingga container pada jaringan ini dapat berkomunikasi satu sama lain.

Anda dapat menginspect jaringan bridge untuk melihat semua containerAnda dapat memeriksa jaringan bridge untuk melihat semua container yang menjadi bagian darinya.

```
agus@agus-Modern-14-C11M:~$ docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "983f9fcda3b63531d0bf1f1fab0ae97dff04cc4767627d6e512d221b7f5e0392",
        "Created": "2024-11-09T02:46:59.067115793+07:00",
        "Scope": "local",
        "Driver": "bridge",
        "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": {
            "195a4ba3022c0e736fb2a9d2c2141fa4c30dcce5e4f56e47aafb5b2f2d022a3f": {
                "Name": "vibrant_einstein",
                "EndpointID": "ee2b50bd5f401cea14334ed777833ac0aeaec3034322b8f21acd95cb7b60e501",
                "MacAddress": "02:42:ac:11:00:06",
                "IPv4Address": "172.17.0.6/16",
                "IPv6Address": ""
            },
            "1acb09631f656aab4696db0f4bb2a73cfd2f13ebd176c8a559297a352ecf7843": {
                "Name": "elegant_lederberg",
                "EndpointID": "98eb930880300806377f0a1d6198bfbbc88a70510d100b046b6f6898b58067ef",
                "MacAddress": "02:42:ac:11:00:03",
                "IPv4Address": "172.17.0.3/16",
                "IPv6Address": ""
            },
            "440e689cd3f6ad9248a30c90ba2c468e5c4efd810570cbabe9c3bb6ba6759043": {
                "Name": "xenodochial_shirley",
                "EndpointID": "aa6230c056de2fb75d5f31b3146d5e7588df2f1feb57eea39eadde62ceeeb99f",
                "MacAddress": "02:42:ac:11:00:04",
                "IPv4Address": "172.17.0.4/16",
                "IPv6Address": ""
            },
            "c466bfedfa2fa5b5a9d6a453a93894f49aa36d4f30ad4c4fc8d8d493ec802c74": {
                "Name": "serene_hopper",
                "EndpointID": "038600438aea83928075e62bd5bc64066be9d92a7aedc94109392a55f772bff1",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            },
            "c5db0a08096edf720e91ad07ca1075883517a32b1ddde88bea53d6536ad03886": {
                "Name": "pedantic_yonath",
                "EndpointID": "e92cf70c8431b01dbf99020935d21b9a315cf4ebb1d4d8495a7aea503b472ee8",
                "MacAddress": "02:42:ac:11:00:05",
                "IPv4Address": "172.17.0.5/16",
                "IPv6Address": ""
            },
            "d680d1e7181a525b1e91a54a31498adbc0390325bdf8539e19c343ae7ae3612b": {
                "Name": "hardcore_chatelet",
                "EndpointID": "614d84b280a33022f707ca760726f4cf203567c8ebf7b5906c030d1f1d524ae9",
                "MacAddress": "02:42:ac:11:00:07",
                "IPv4Address": "172.17.0.7/16",
                "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": {}
    }
]
agus@agus-Modern-14-C11M:~$
```

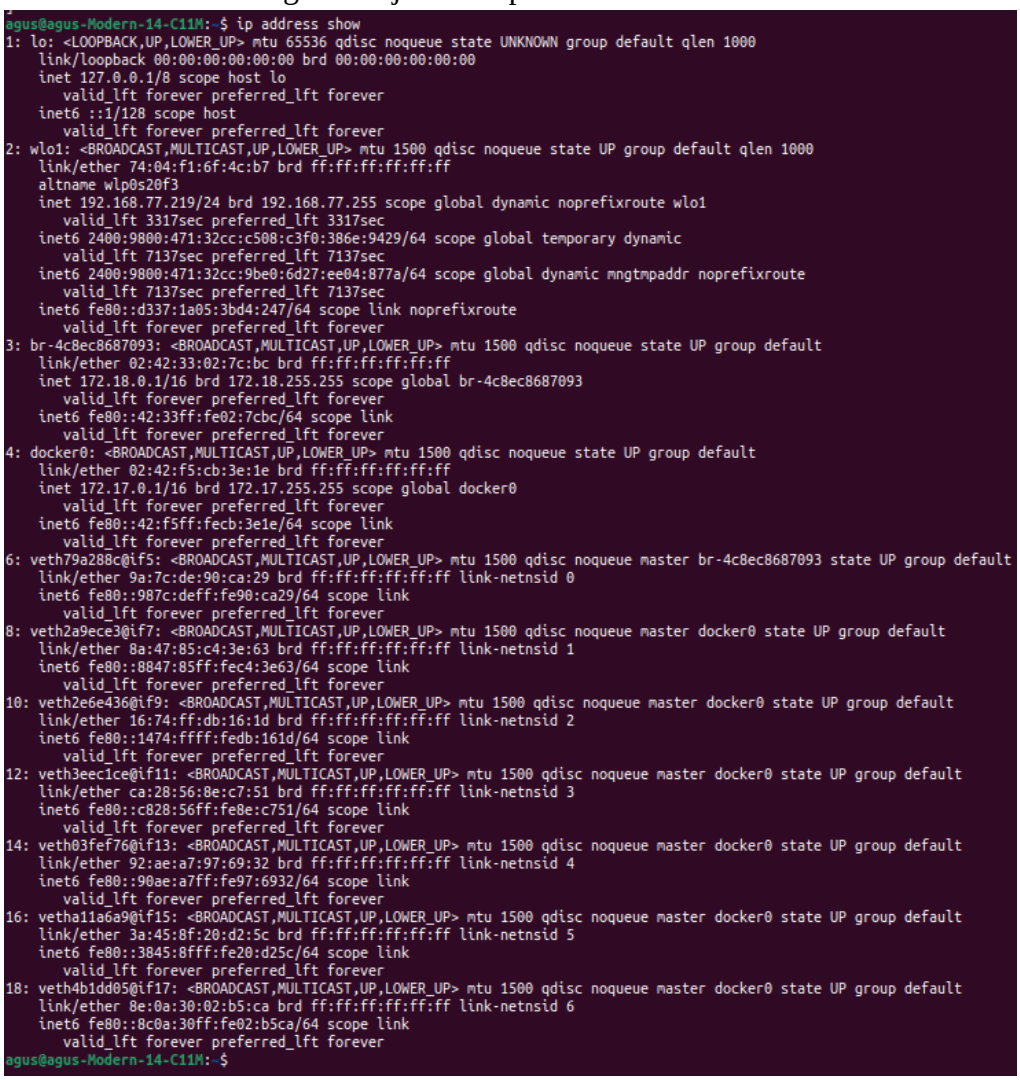
Anda dapat melihat container apache dan nginx sebagai bagian dari jaringan bridge default.

Kita juga akan melihat bahwa container menggunakan IP yang berbeda dari host.

Kita akan memverifikasi di bagian selanjutnya.

1. Alamat IP Host dan docker0

   docker0 adalah jaringan yang dikonfigurasi pada host itu sendiri. Anda dapat memverifikasi ini dengan menjalankan perintah berikut.



   Di sini kita dapat melihat dengan jelas bahwa host memiliki IP yang berbeda dengan IP container nginx yang kita buat dan docker0.

2. Membuat jaringan kita sendiri

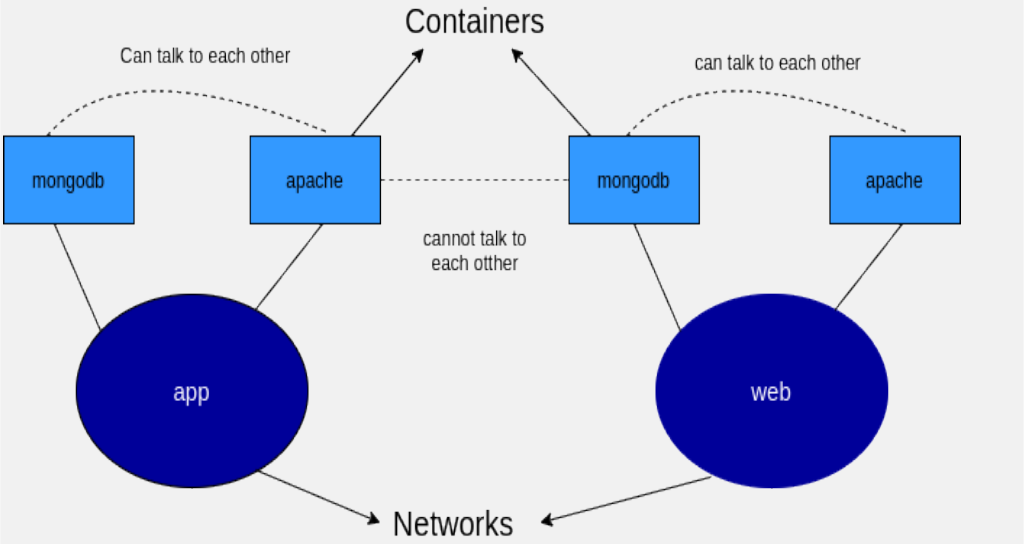
   Sekarang, kita telah mengetahui bahwa container dapat berbicara satu sama lain pada jaringan yang sama.

   Mari kita pertimbangkan skenario di mana kita memiliki dua aplikasi berbeda yang berjalan, yang tidak terkait satu sama lain. Bagaimana kita memastikan tidak ada komunikasi yang diizinkan di antara container dari aplikasi yang berbeda dalam skenario seperti itu?

   Mari kita lihat solusinya.

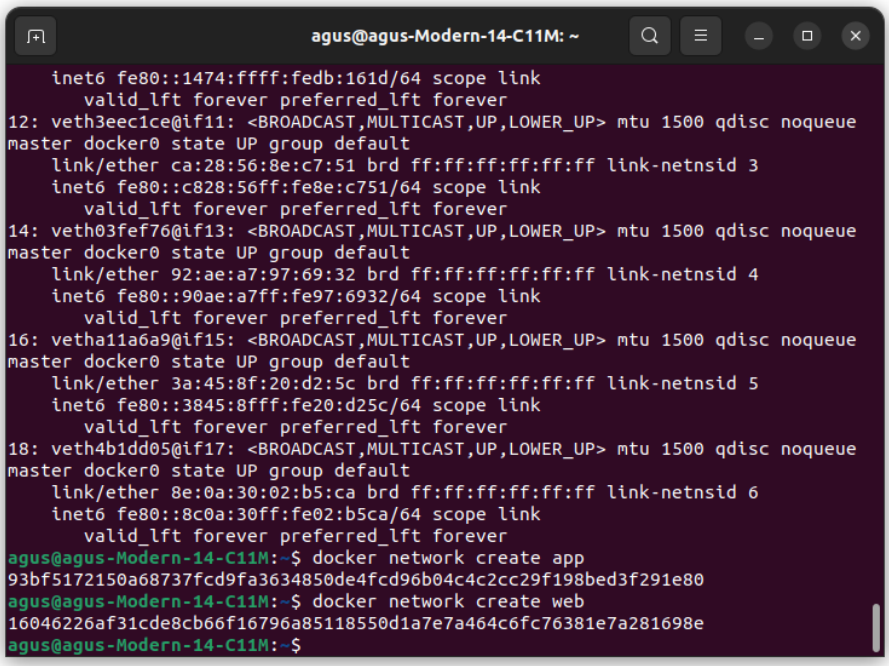
   Kita dapat membuat jaringan sendiri untuk mengisolasi container secara logis.

Kontainer dalam jaringan yang sama dapat berbicara satu sama lain.

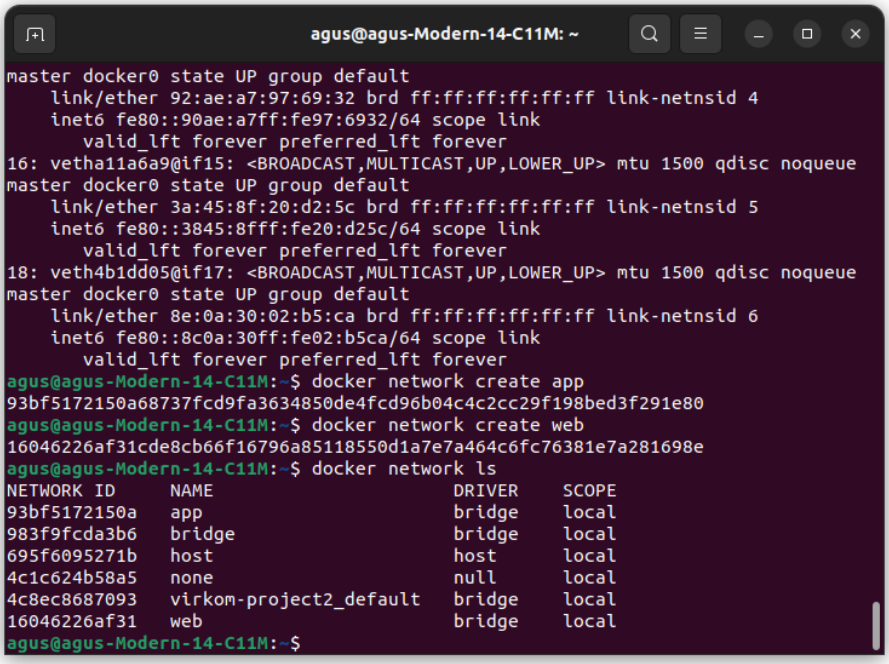


Mari kita lihat ini dalam tindakan.

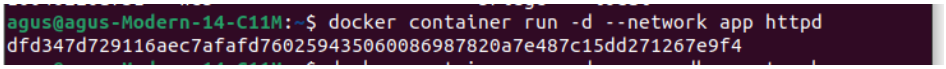
1. Buat jaringan yang disebut aplikasi dan jaringan lain yang disebut web.



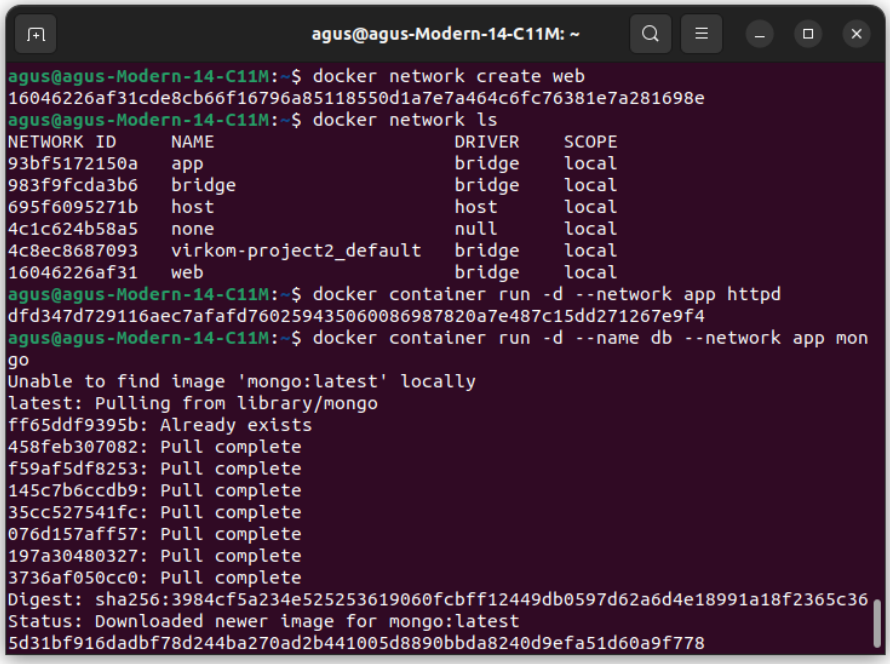
2. Mari kita verifikasi bahwa jaringan telah dibuat.



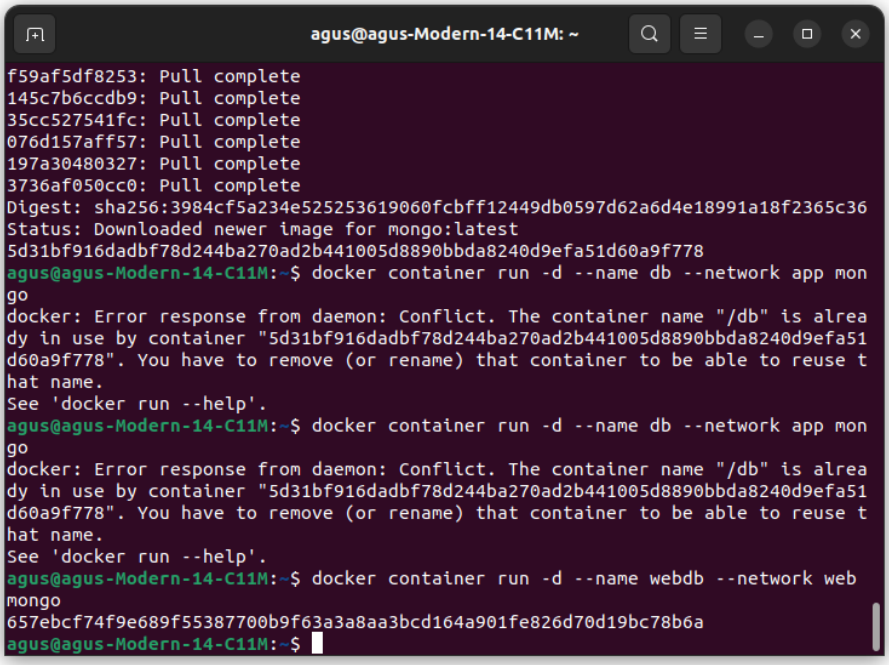
3. Mulai container apache dan menambahkannya ke jaringan aplikasi.



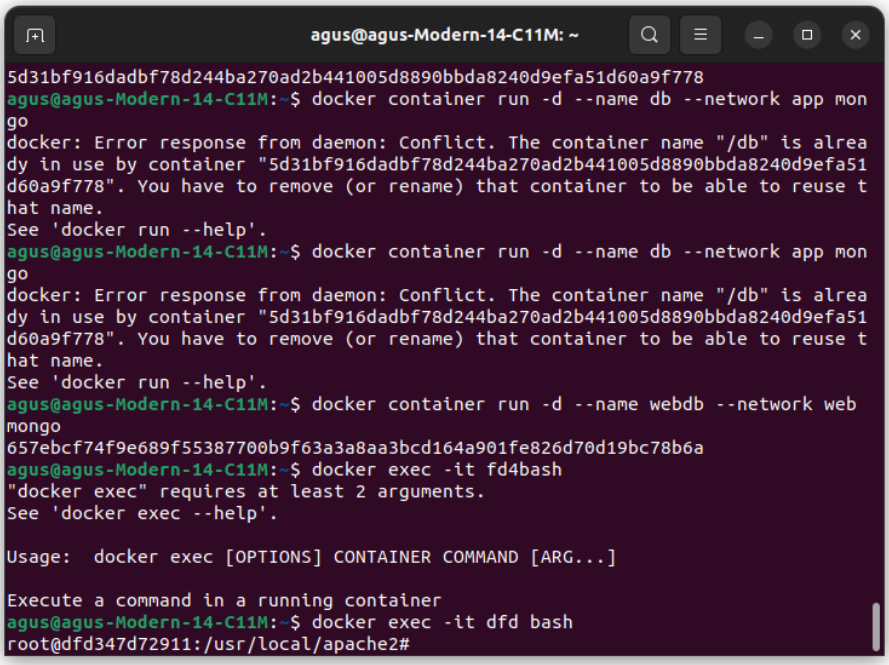
4. Mulai container mongo db dan tambahkan juga ke jaringan aplikasi.



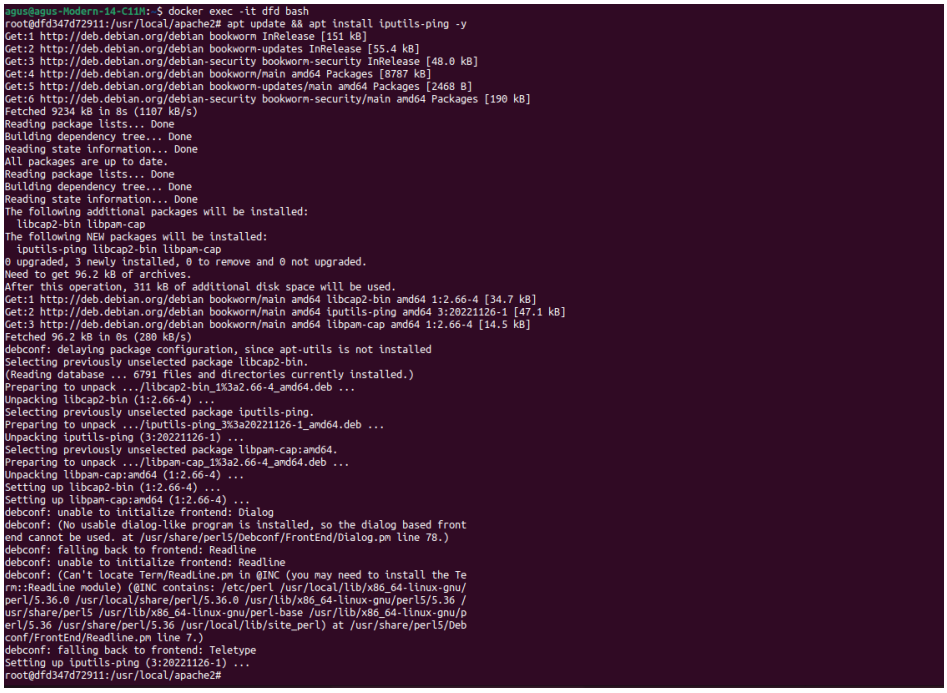
5. Mulai container mongo db yang lain dan tambahkan ke jaringan web sebagai berikut.



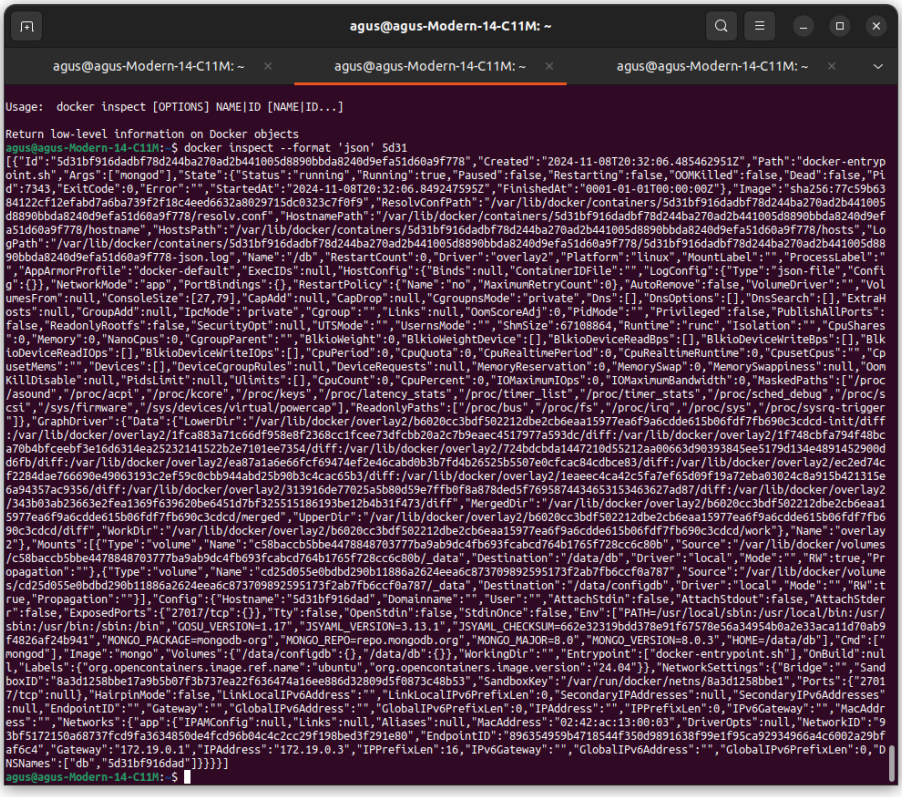
6. Memulai sesi bash di dalam container apache.



7. Instal IPutils-ping untuk melakukan ping untuk berbicara dengan container mongo.

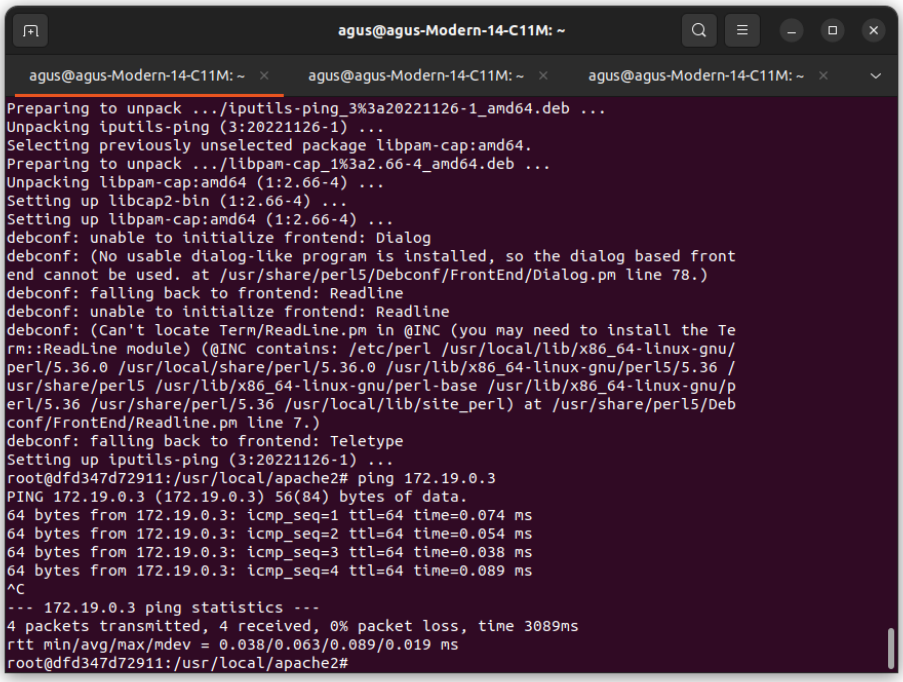


8. Cari tahu alamat IP dari db container.

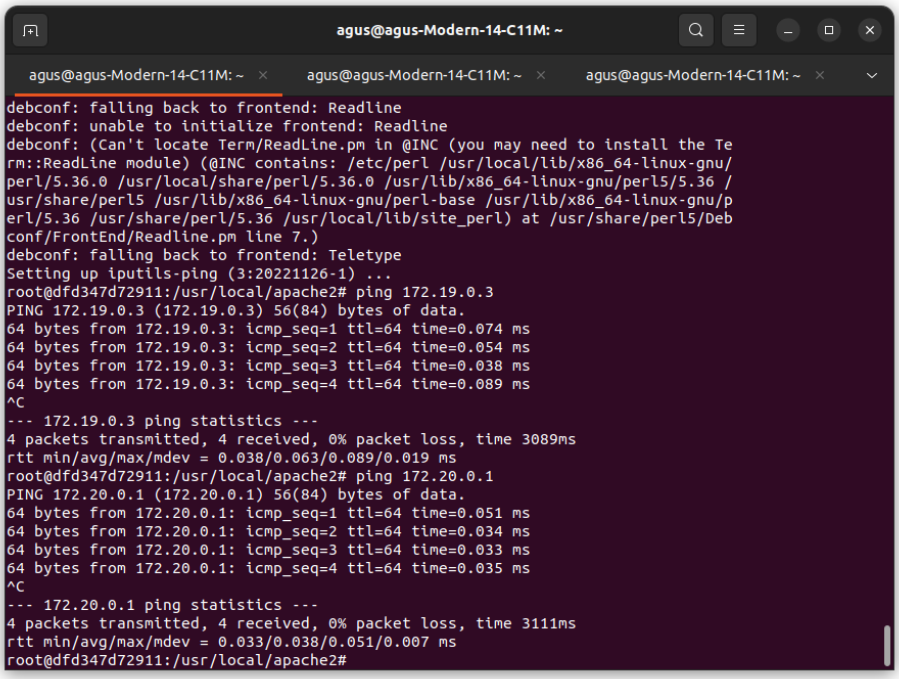


Hebat! kita dapat berbicara dengan container mongo di jaringan yang sama.

9. Ping mongo container menggunakan alamat IP yang kita lihat di atas.



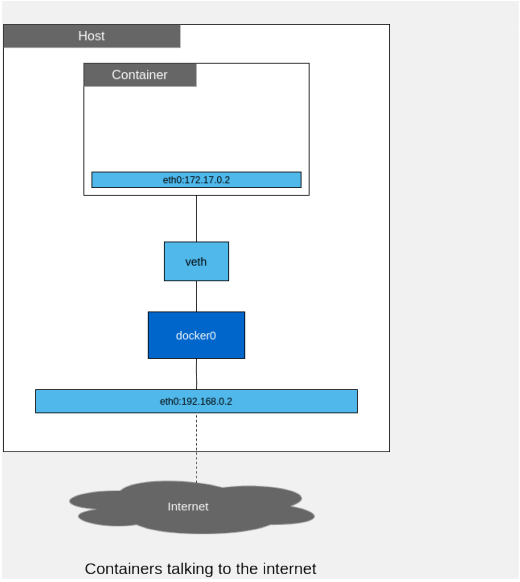
10. Sekarang, mari kita coba melakukan ping ke container webdb dari jaringan lain. Cari tahu alamat IP dengan cara yang sama yang sama seperti yang kita lakukan sebelumnya.



Dengan demikian, kami dapat memastikan container dari aplikasi yang berbeda tidak dapat berbicara satu sama lain.

Info Bonus: Wadah di jaringan yang berbeda tidak dapat berkomunikasi satu sama lain karena mereka berada di subnet yang berbeda.

- **Bagaimana cara container berbicara dengan internet?**



Containers talking to the internet

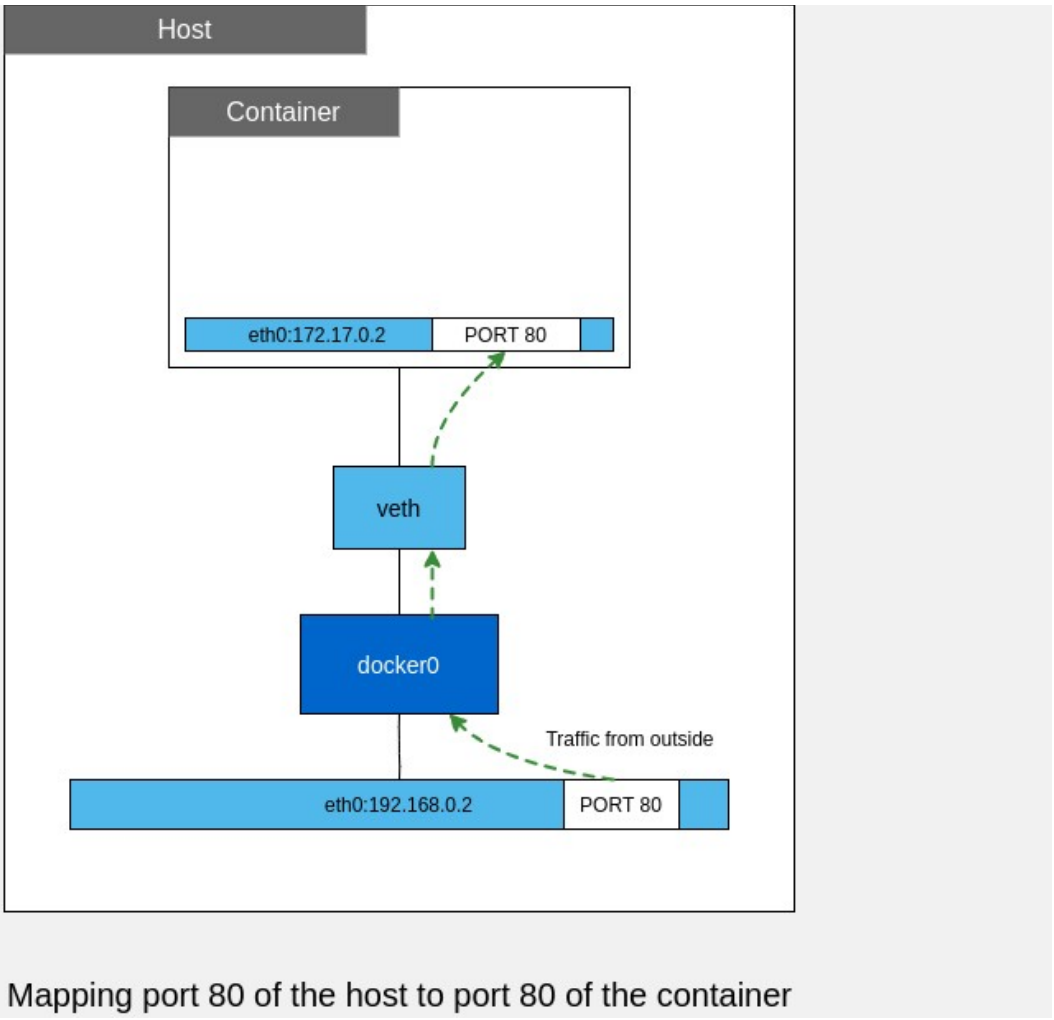
Kontainer berbicara dengan jaringan docker0/jaringan yang ditentukan pengguna menggunakan IP yang ditetapkan oleh jaringan ini.

Jaringan docker0/jaringan bridge menggunakan IP host pada antarmuka defaultnya untuk berbicara dengan internet.

Jaringan menjaga agar container benar-benar terisolasi dari apa pun yang tidak berada di jaringan yang sama.

- **Lalu, bagaimana dunia luar berinteraksi dengan container tersebut?**

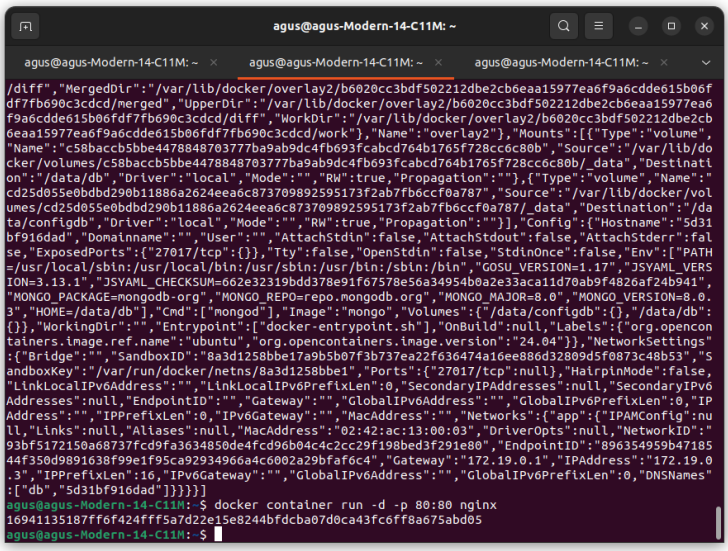
  Hal ini dapat dicapai dengan membuka port pada container dan memetakannya ke port host.



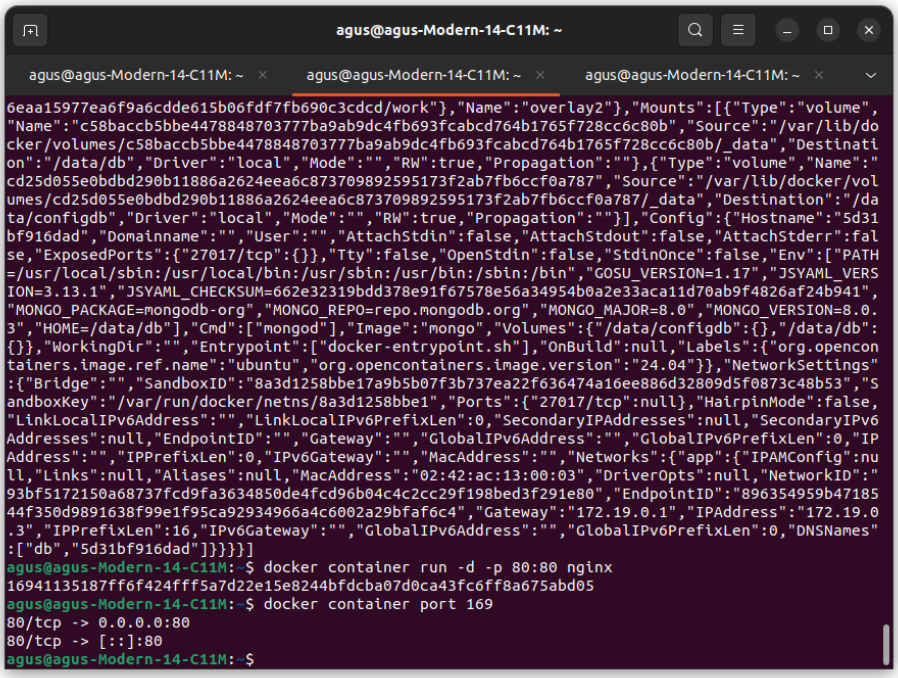
Mapping port 80 of the host to port 80 of the container

Mari kita lihat bagaimana cara melakukannya.

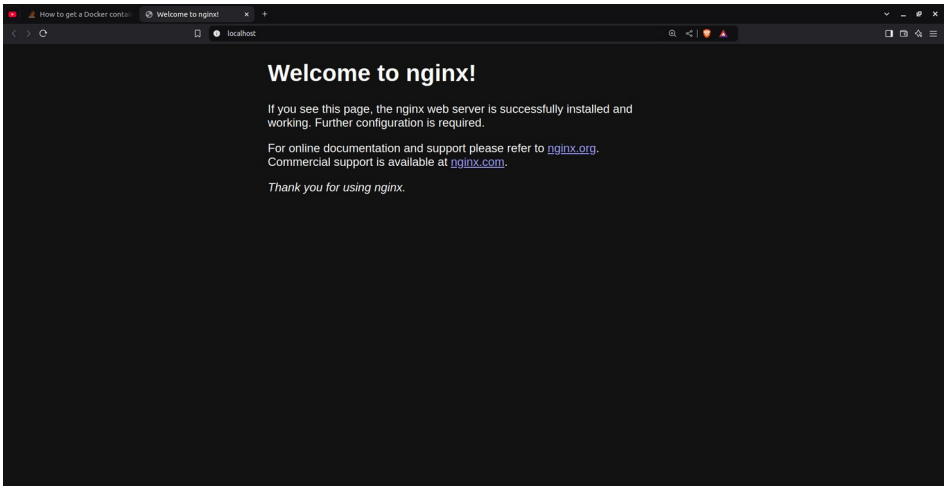
1. Jalankan nginx container dengan dan petakan port 80 host ke port 80 container.



2. Mari kita lihat apakah pemetaan berhasil dilakukan.



3. Mari kita juga memverifikasi apakah kita dapat melihat server nginx yang berjalan pada port 80 host dengan menavigasi ke http://localhost:80



- **Penamaan DNS**

  Dalam lingkungan docker, alamat IP tidak dapat diandalkan.

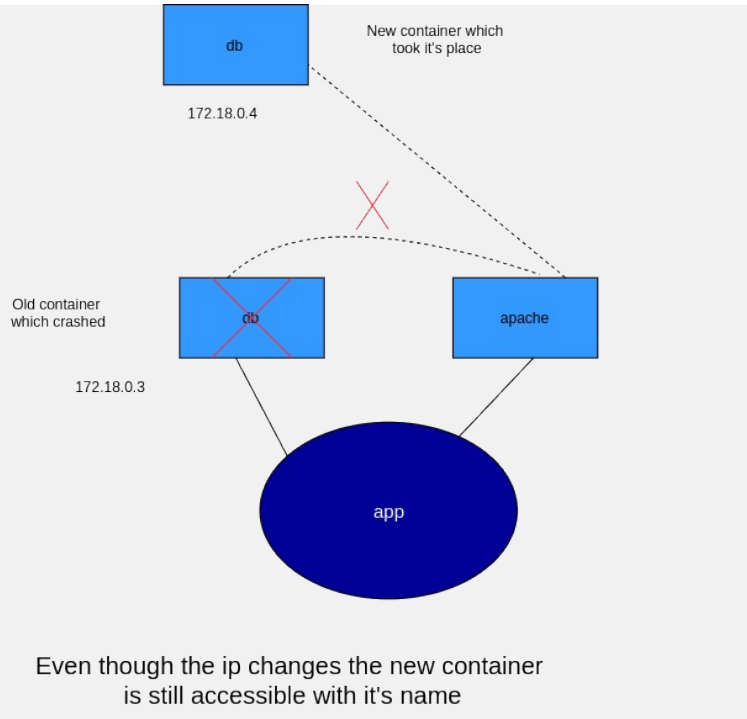
  Ini karena container dapat macet/berhenti secara tidak terduga dan kemudian container baru menggantikannya dengan IP yang berbeda.

  Jadi, bagaimana kita memastikan komunikasi dalam situasi seperti itu?

Di sinilah penamaan DNS berperan. Docker secara default menggunakan nama container sebagai nama host yang dapat digunakan oleh container untuk berbicara satu sama lain.

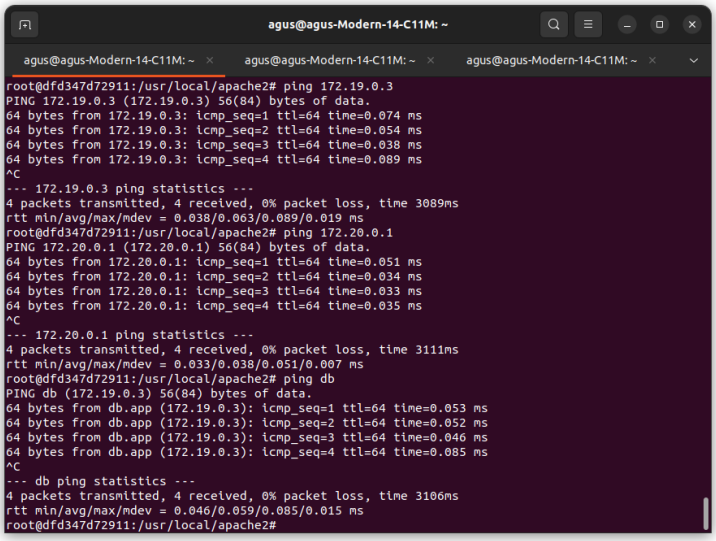
Catatan: Ini hanya berlaku untuk jaringan yang dibuat oleh pengguna. Jaringan bridge default tidak memiliki fitur ini.

Baca artikel ini untuk mengetahui perbedaan antara jaringan yang ditentukan pengguna dan jaringan default.

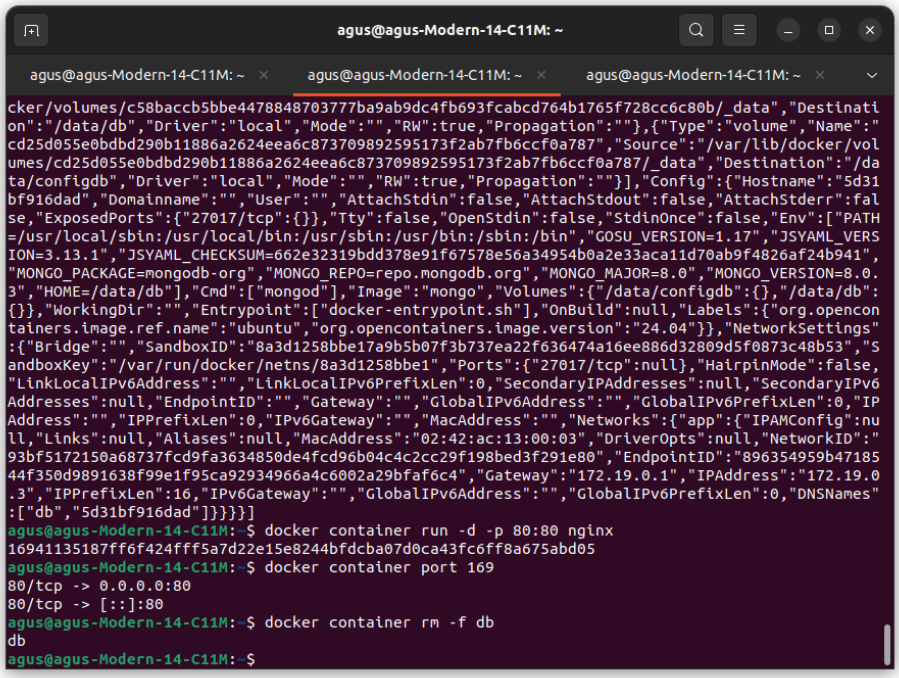


Mari kita lakukan latihan untuk melihat bagaimana hal ini bermanfaat.

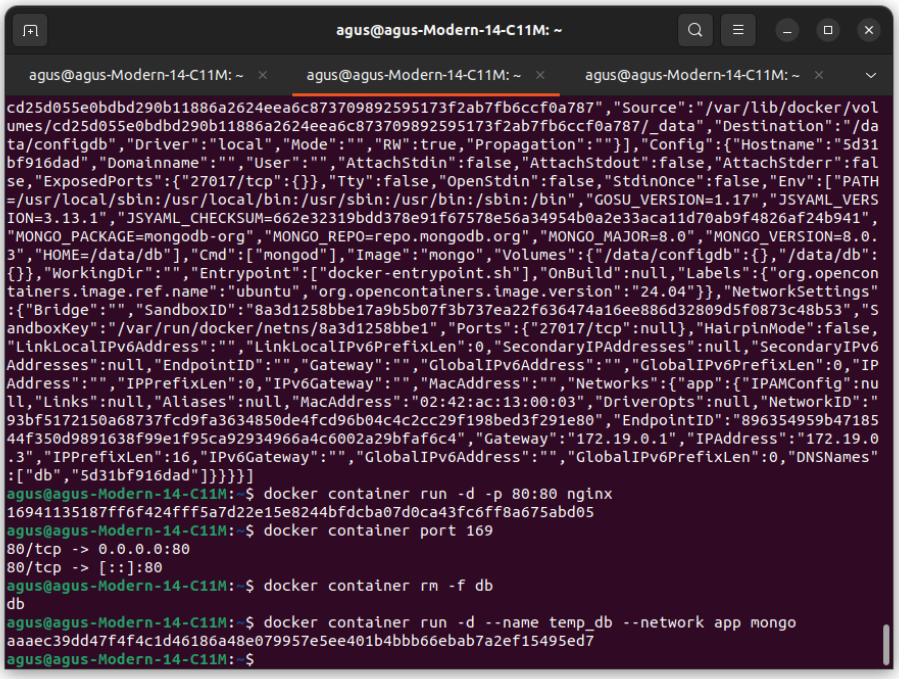
1. Dalam sesi bash yang sama, ping container mongo menggunakan namanya.



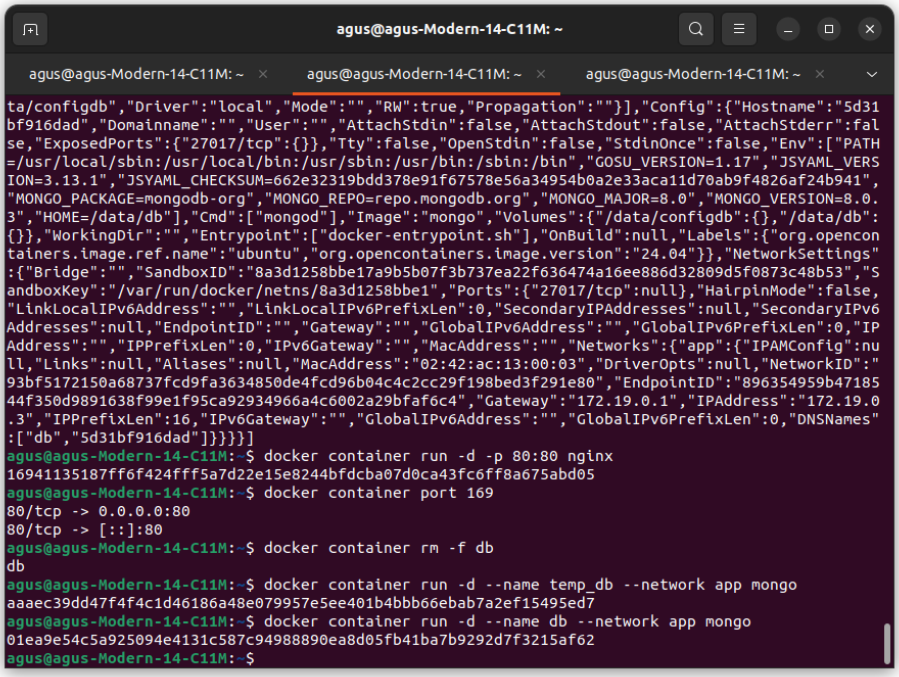
2. Hentikan dan hapus container ini di tab yang berbeda.



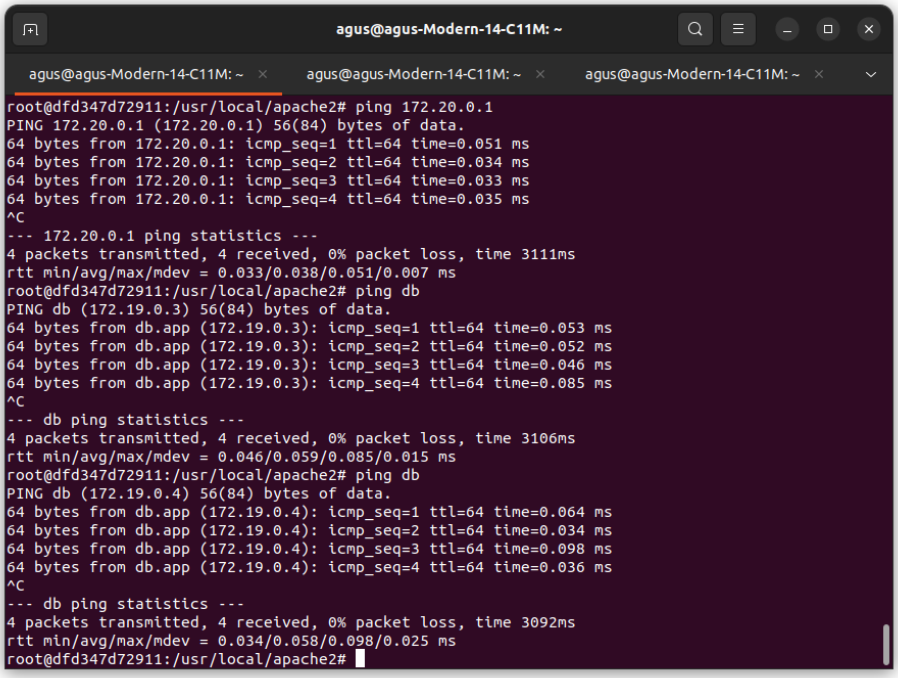
3. Mulai container apa pun di jaringan yang sama sehingga IP 172.20.0.3 dialokasikan untuknya.



4. Memulai container mongo lain dengan nama yang sama.



5. Coba ping container db lagi dari container apache.



Anda akan melihat bahwa meskIPun IP container telah berubah , kami dapat melakukan ping menggunakan DNS (nama container).

Hal ini memastikan komunikasi yang andal dalam lingkungan yang sangat dinamis.

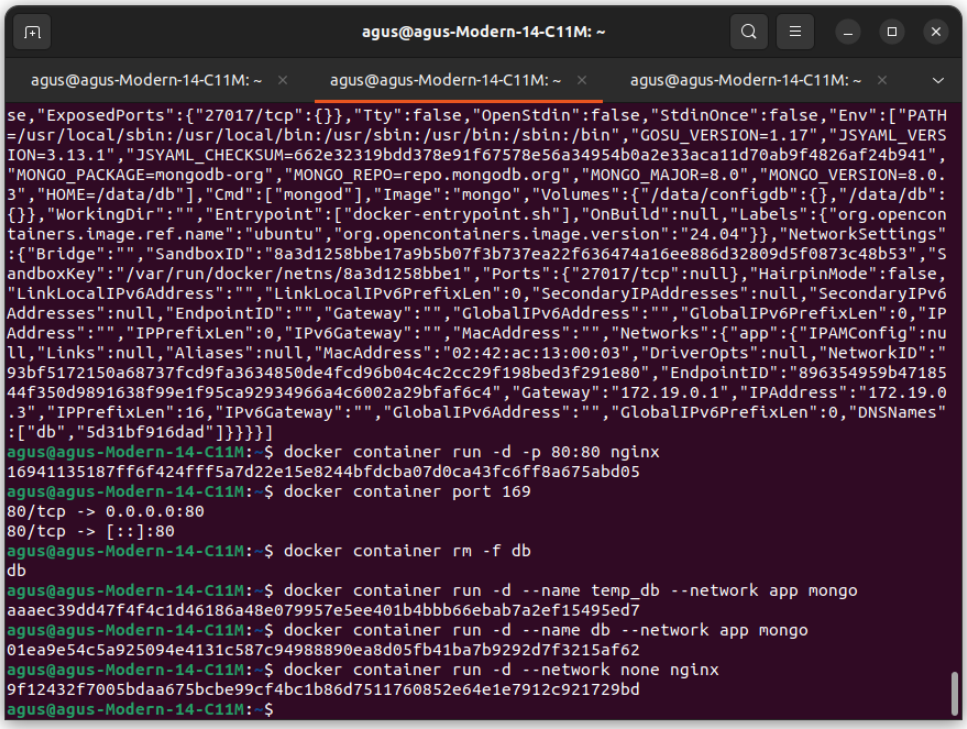
Catatan : Pada jaringan docker0 untuk berkomunikasi menggunakan DNS, kita harus menggunakan flag tautan saat membuat container yang sudah tidak digunakan lagi dan tidak boleh digunakan. Anda sebaiknya membuat jaringan baru.

Baca artikel ini untuk mengetahui lebih lanjut tentang hal ini.

- **Jaringan None**

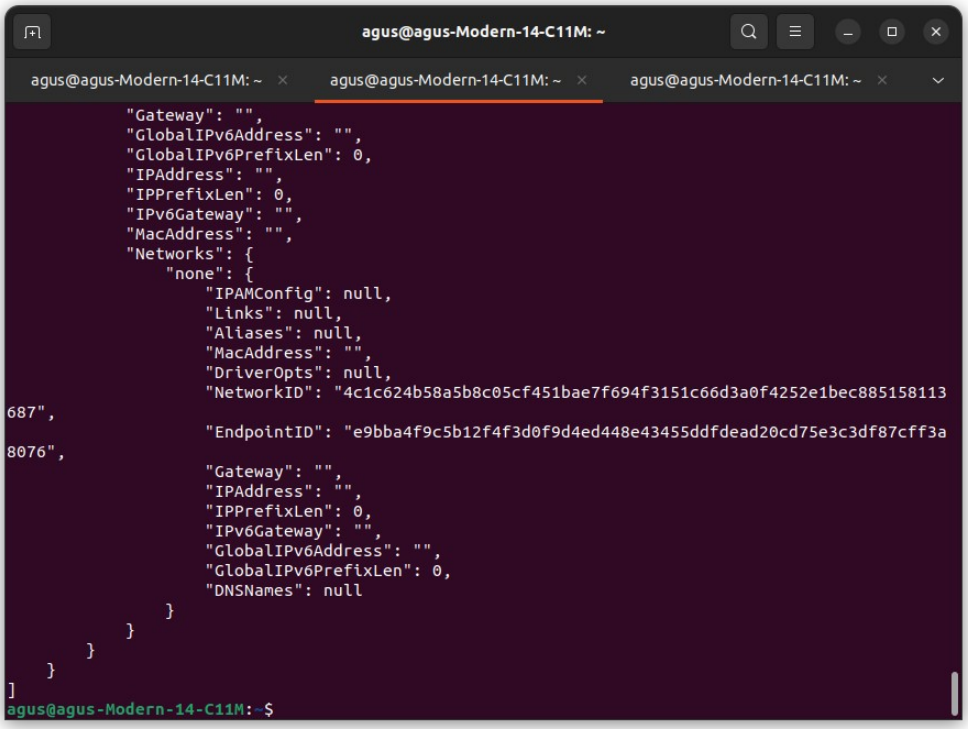
  Ketika Anda menentukan container untuk tidak menggunakan jaringan, maka container tersebut akan sepenuhnya terisolasi. Ia tidak akan dapat berbicara dengan container lain atau internet.

  Tidak ada alamat IP yang diberikan ke container tersebut.



Mari kita coba mencatat alamat IP dari container ini.

```
agus@agus-Modern-14-C11M:~$ docker container inspect 9f12
```



Anda akan melihat tidak ada alamat IP yang ditetapkan untuk container ini.

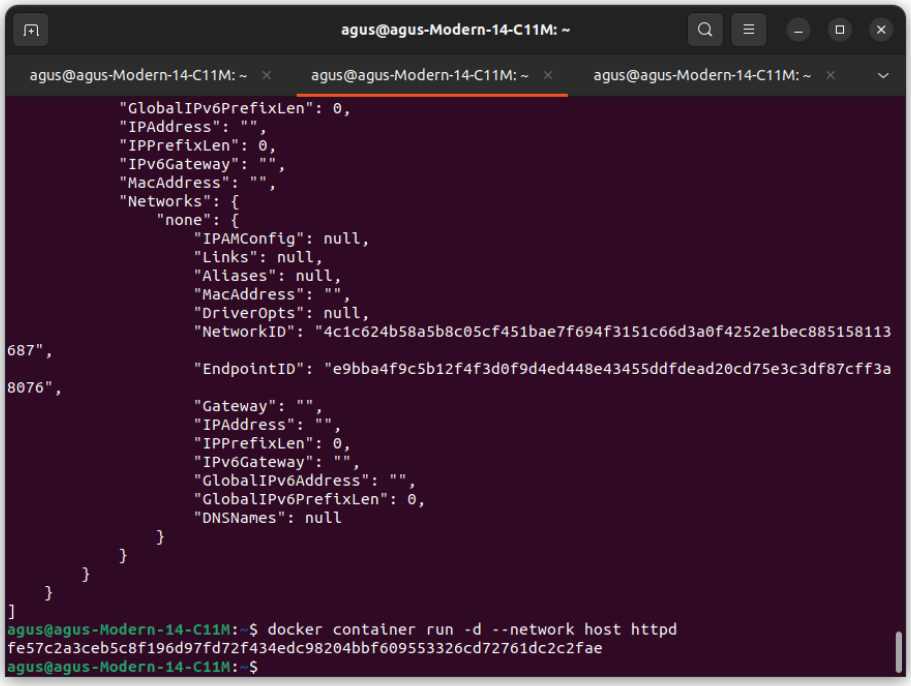
- **Jaringan host**

  Terkadang ada kebutuhan untuk menggunakan jaringan host itu sendiri. Dalam hal ini Anda kehilangan beberapa manfaat containerisasi tetapi jaringan menjadi cepat.

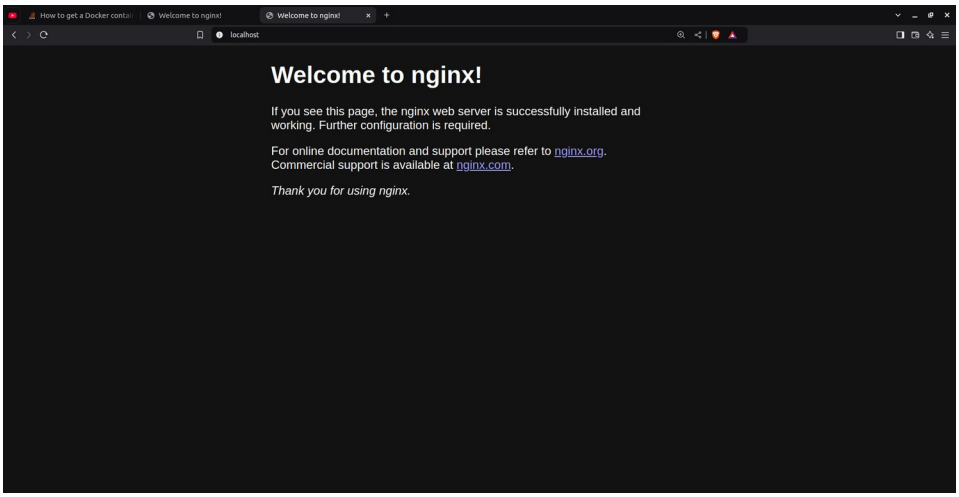
  Dalam hal ini tidak ada jaringan baru atau alamat IP baru yang ditetapkan. Kontainer menggunakan IP host itu sendiri.

  Catatan: Driver jaringan host hanya bekerja pada host Linux, dan tidak didukung pada Docker Desktop untuk Mac, Docker Desktop untuk Windows, atau Docker EE untuk Windows Server.

Mari kita verifikasi hal ini.



Navigasikan ke http:localhost.



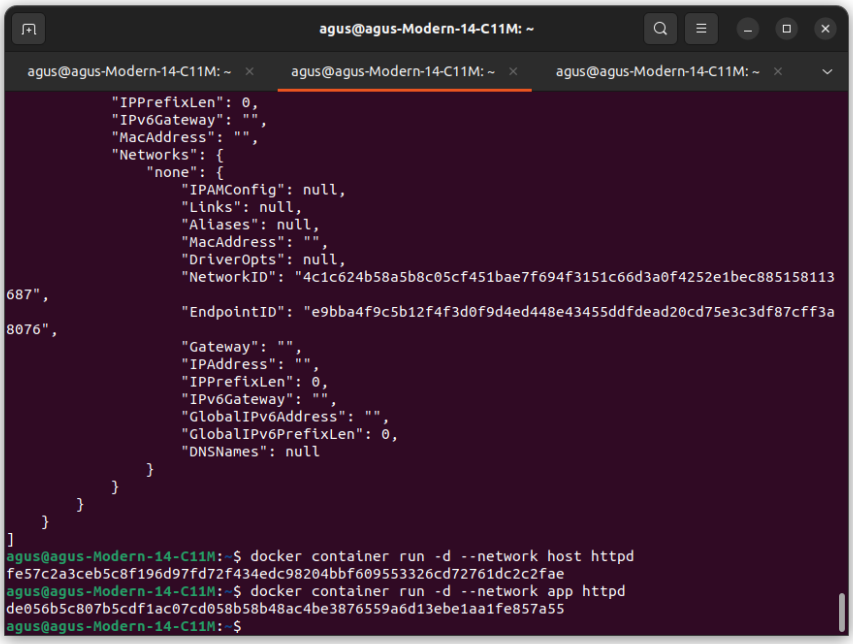
Anda akan melihat bahwa server apache berjalan pada port host 80 tanpa mempublikasikan port ini. Ini karena container menggunakan jaringan host itu sendiri.

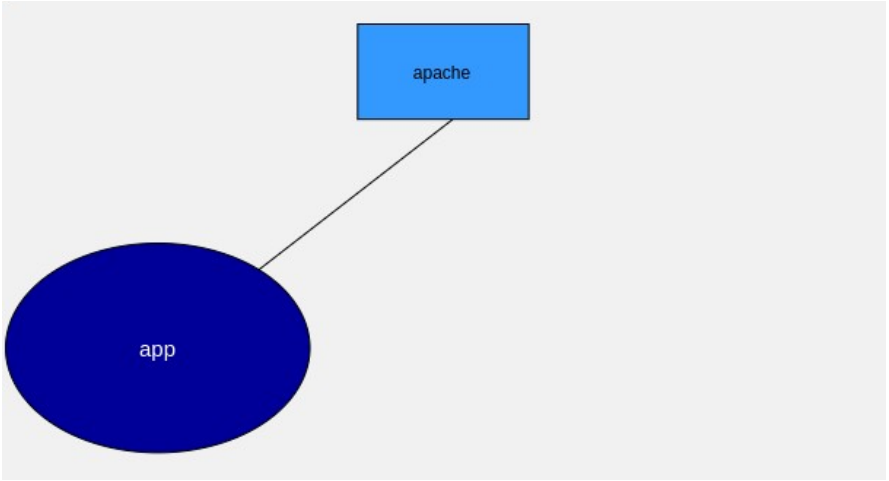
- **Menambahkan dan menghapus container ke jaringan secara dinamis.**

  Kontainer dapat dIPasang dan dihapus dari jaringan mana pun secara dinamis.

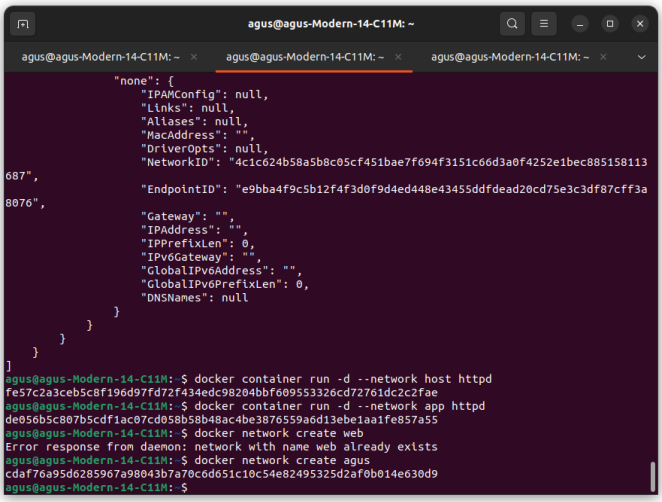
  Mari kita lihat bagaimana cara melakukannya.

1. Jalankan container apache dan lampirkan ke jaringan aplikasi.

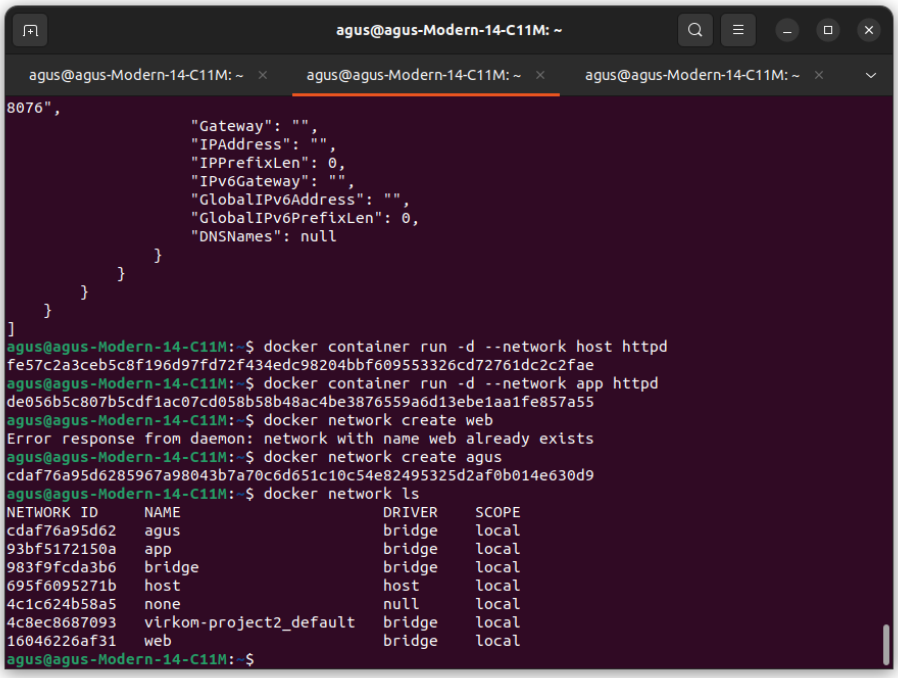




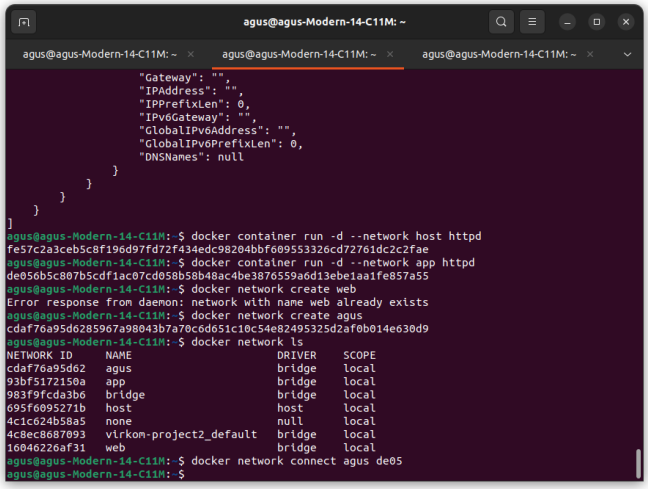
2. Buat jaringan baru dengan nama <web>.

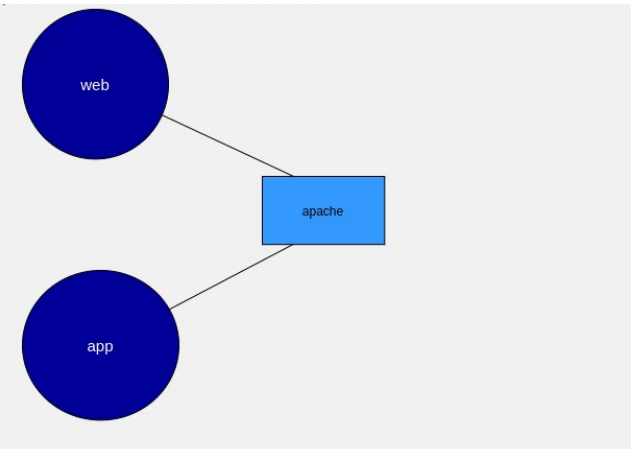


3. Verifikasi bahwa jaringan telah dibuat.



4. Memasang container apache ke jaringan web.





5. Verifikasi bahwa container terpasang ke kedua jaringan.

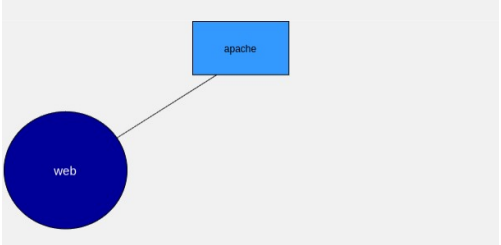
```
agus@agus-Modern-14-C11M:~$ docker container inspect de05
```


```
        "Networks": {
            "agus": {
                "IPAMConfig": {},
                "Links": null,
                "Aliases": [],
                "MacAddress": "02:42:ac:15:00:02",
                "DriverOpts": {},
                "NetworkID": "cdaf76a95d6285967a98043b7a70c6d651c10c54e82495325d2af0b014e630d9",
                "EndpointID": "8cbac2da7bc44f5563ef74565b13d748f0f67139f955d2e57af3ab076f8181dc",
                "Gateway": "172.21.0.1",
                "IPAddress": "172.21.0.2",
                "IPPrefixLen": 16,
                "IPv6Gateway": "",
                "GlobalIPv6Address": "",
                "GlobalIPv6PrefixLen": 0,
                "DNSNames": [
                    "relaxed_cannon",
                    "de056b5c807b"
                ]
            },
            "app": {
                "IPAMConfig": null,
                "Links": null,
                "Aliases": null,
                "MacAddress": "02:42:ac:13:00:05",
                "DriverOpts": null,
                "NetworkID": "93bf5172150a68737fcd9fa3634850de4fcd96b04c4c2cc29f198bed3f291e80",
                "EndpointID": "d1b13f62be832bb6e40e8d751d85c109754d1c268f74e4abf0055221310eafc9",
                "Gateway": "172.19.0.1",
                "IPAddress": "172.19.0.5",
                "IPPrefixLen": 16,
                "IPv6Gateway": "",
                "GlobalIPv6Address": "",
                "GlobalIPv6PrefixLen": 0,
                "DNSNames": [
                    "relaxed_cannon",
                    "de056b5c807b"
                ]
            }
        }
    }
]
```


Anda akan melihat container container yang sama berada di dua jaringan yang berbeda dan memiliki dua alamat IP yang berbeda.

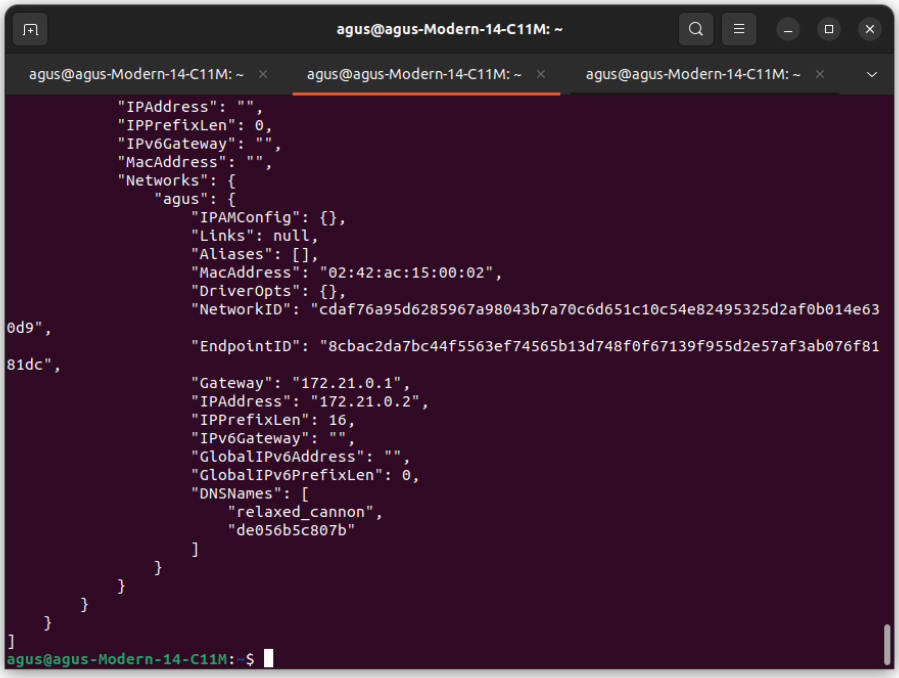
6. Mari hapus container dari jaringan aplikasi sekarang


```
agus@agus-Modern-14-C11M: ~
agus@agus-Modern-14-C11M: ~   agus@agus-Modern-14-C11M: ~   agus@agus-Modern-14-C11M: ~
                    "relaxed_cannon",
                    "de056b5c807b"
                ]
            },
            "app": {
                "IPAMConfig": null,
                "Links": null,
                "Aliases": null,
                "MacAddress": "02:42:ac:13:00:05",
                "DriverOpts": null,
                "NetworkID": "93bf5172150a68737fcd9fa3634850de4fcd96b04c4c2cc29f198bed3f291
e80",
                "EndpointID": "d1b13f62be832bb6e40e8d751d85c109754d1c268f74e4abf0055221310e
afc9",
                "Gateway": "172.19.0.1",
                "IPAddress": "172.19.0.5",
                "IPPrefixLen": 16,
                "IPv6Gateway": "",
                "GlobalIPv6Address": "",
                "GlobalIPv6PrefixLen": 0,
                "DNSNames": [
                    "relaxed_cannon",
                    "de056b5c807b"
                ]
            }
        }
    }
]
agus@agus-Modern-14-C11M:~$ docker network disconnect app de05
agus@agus-Modern-14-C11M:~$
```

7. Mari kita verifikasi bahwa aplikasi tersebut telah dihapus dari jaringan aplikasi.

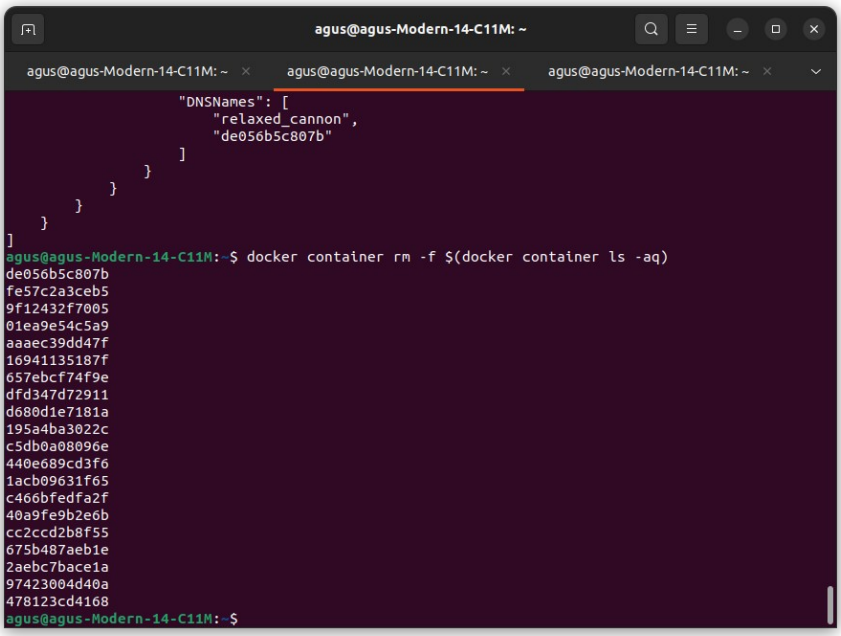
```
agus@agus-Modern-14-C11M:~$ docker container inspect de05
```



- **Menghapus jaringan yang ada**

  Sebagai langkah terakhir, mari kita hapus semua container dan hapus semua jaringan yang telah kita buat.

  1. Keluarkan semua container sekaligus.



2. Hapus semua jaringan yang kita buat.