

NETWORK

MODUL PCC CLASS

"Introduction Container and Docker"



Our Sponsorship :

Supported by :

More Information :

PCC CLASS

DEPARTEMEN NETWORK

Introduction Container and Docker

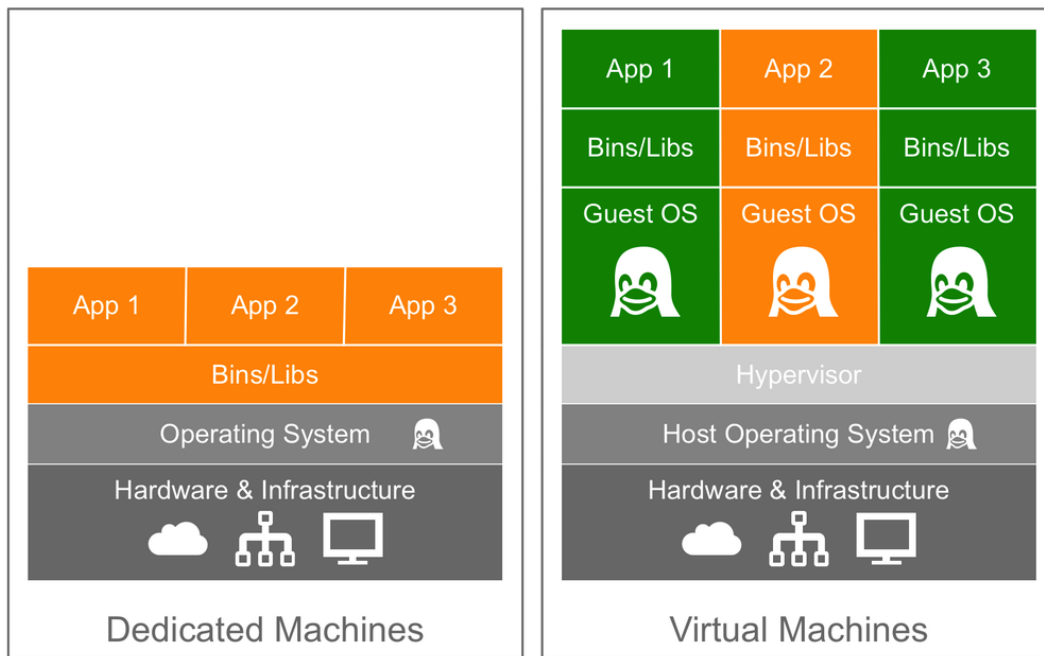
BAB I

Dasar Teori

1.1 CONTAINER

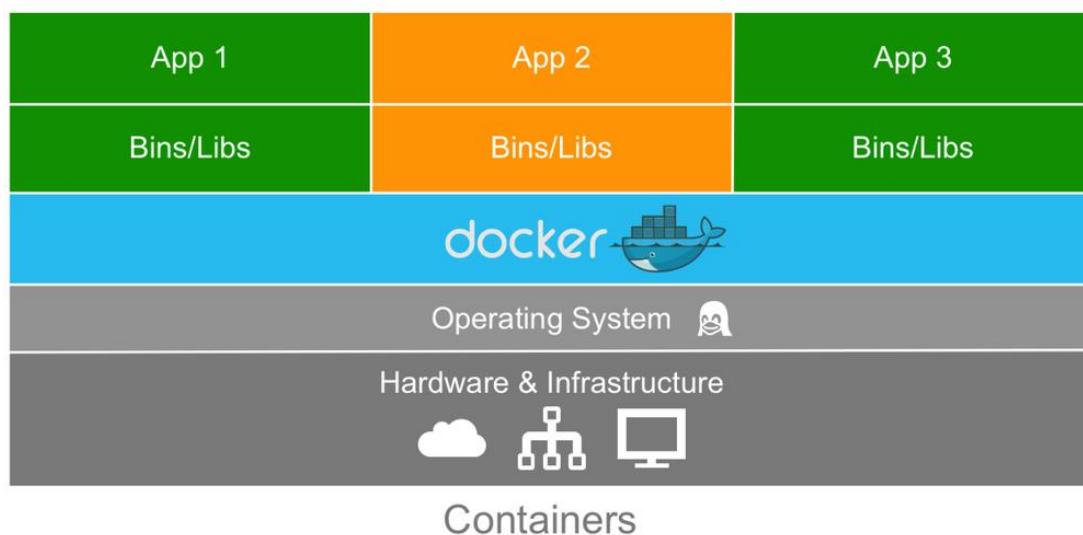
1.1.1 Pengertian Container

Pada pengembangan aplikasi secara konvensional, developer akan menyerahkan aplikasi yang telah selesai kepada bagian Operation untuk diinstall di server produksi, hal ini akan menjadi sulit jika aplikasi tersebut merasa bergantung pada beberapa kebutuhan seperti versi, library, dan depedensi yang lain. Untuk mengatasi tantangan yang ditimbulkan oleh beragam aplikasi dan kebutuhannya tersebut, maka salah satu solusinya adalah menggunakan Vitual Machine (VM) untuk mengisolasi aplikasi. Pada saat kita menjalankan VM, kita harus menginstall sistem operasinya terlebih dahulu. Misal kita membuat aplikasi yang lumayan banyak, mau tidak mau kita harus membuat VM yang lebih banyak juga. Dan tentu saja hal tersebut akan membutuhkan resource yang lebih banyak. Jadi pada saat kita ingin membuat VM lebih banyak, kita akan mengulangi proses instalasi sampai VM itu sesuai dengan kebutuhan, dan tentu itu pekerjaan yang melelahkan dan memakan banyak waktu.



sumber: ADINUSA (Akademi Digital Nusantara)

Solusi lainnya adalah menggunakan Container. Berbeda dengan VM, Container sendiri berfokus pada sisi Aplikasi. Container sendiri sebenarnya berjalan diatas aplikasi Container Manager yang berjalan di sistem operasi, salah satu contoh conainer manager adalah Docker. Docker Container yang dapat meng-enkapsulasi ringan untuk aplikasi dan kebutuhannya tanpa harus menginstall sistem operasi tersendiri. Container memungkinkan developer untuk mengemas aplikasi, aplikasi dan dependency lainnya ke dalam unit yang terisolasi, sehingga memudahkan penerapannya di berbagai lingkungan.



sumber: ADINUSA (Akademi Digital Nusantara)

Container akan menggunakan sistem operasi host dimana Container managernya berjalan. Oleh karena itu, Container akan lebih hemat resource dan lebih cepat, karena tidak butuh sistem operasi tersendiri. Container ini dapat meminimalkan penggunaan resource pada perangkat lunak dengan lebih konsisten dan portable, yang artinya pengembang dapat "membangun, mengirim, dan menjalankannya di mana saja," sehingga mendukung siklus deploy atau rilis aplikasi menjadi lebih singkat.

1.2 DOCKER

1.2.1 Pengertian Docker

Docker merupakan salah satu platform Container Manager yang populer. Docker memungkinkan developer untuk mengemas, mengirim, dan menjalankan aplikasi beserta dependensinya dalam lingkungan yang terisolasi yang disebut sebagai "container.". Sebagai container manager, Docker menyediakan berbagai fitur dan fungsionalitas yang memungkinkan pengguna untuk membuat, mengelola, dan menjalankan kontainer dengan mudah. Container Docker dapat berisi semua yang diperlukan oleh suatu aplikasi untuk berjalan, termasuk kode aplikasi, runtime, library, alat sistem, dan variabel lingkungan. Docker menggunakan teknologi kontainerisasi untuk memastikan bahwa aplikasi dapat berjalan dengan konsisten di berbagai lingkungan, baik itu di lingkungan pengembangan, pengujian, maupun produksi. Docker menyederhanakan proses pengembangan dan distribusi aplikasi dengan memastikan bahwa setiap kontainer berjalan secara independen dan dapat dipindahkan dari satu lingkungan ke lingkungan lain tanpa perlu memodifikasi kode atau konfigurasi aplikasi.

Fitur utama dari Docker sebagai container manager meliputi:

1. **Pembuatan Kontainer:** Docker memungkinkan pengguna untuk membuat kontainer dengan mudah dari image yang telah ditentukan sebelumnya atau melalui file Dockerfile yang berisi instruksi untuk membangun image.
2. **Penyimpanan dan Penyebaran Image:** Docker menyediakan repository pusat yang disebut Docker Hub dimana pengguna dapat menyimpan dan berbagi image Docker. Selain itu, Docker juga memungkinkan pengguna untuk membuat dan menyebarkan image Docker sendiri.

3. **Pengaturan Jaringan:** Docker memungkinkan pengguna untuk mengatur jaringan antar-kontainer atau antara kontainer dan host, termasuk konfigurasi seperti port forwarding dan pengaturan jaringan yang lebih kompleks.
4. **Manajemen Resource:** Docker memberikan kontrol yang baik atas sumber daya yang digunakan oleh kontainer, seperti CPU, memori, dan ruang disk.
5. **Monitoring dan Logging:** Docker menyediakan alat untuk memantau kesehatan dan kinerja kontainer, serta untuk mengumpulkan dan menganalisis log dari kontainer.
6. **Otomatisasi dan Orkestrasi:** Docker dapat diintegrasikan dengan alat-alat orkestrasi seperti Docker Swarm atau Kubernetes untuk otomatisasi pengelolaan kontainer dalam lingkungan yang lebih besar dan kompleks.

Dengan fitur-fitur ini, Docker memungkinkan pengguna untuk dengan mudah membuat, mengelola, dan menjalankan kontainer dengan efisien, menjadikannya pilihan utama sebagai container manager dalam pengembangan dan pengelolaan aplikasi modern.

1.2.2 Docker Registry

Docker Registry adalah repository tempat penyimpanan untuk Docker images. Ini berfungsi sebagai tempat sentral untuk menyimpan dan mengelola images Docker. Registry dapat bersifat publik atau privat, tergantung pada kebutuhan pengguna. Beberapa contoh Docker Registry yang populer termasuk Docker Hub (publik) dan AWS Elastic Container Registry (ECR) (privat). Pada Docker Hub sudah tersedia banyak images dari stack-stack terkenal seperti php, mysql, nginx, python, go, dll.

Sistem Registry di Docker hampir sama dengan sistem git, dimana di Registry kita bisa pull (download image orang lain / dari suatu sumber) atau kita juga bisa push (membuat image kita sendiri lalu menguploadnya).

1.2.3 Docker Images

Pada Virtual Machine, images hampir sama dengan “Sistem Operasi” yang akan diinstall. Perbedaanya, Docker Images merupakan sebuah hasil building/installer aplikasinya yang berisi aplikasi siap digunakan. Artinya apa? Image di docker ini adalah suatu keadaan dimana aplikasi sudah siap dijalankan, jadi bukan seperti Installer Sistem Operasi seperti di Virtual Machine. Seandainya kita memerlukan

aplikasi berbasis linux, kita bisa mendownload image tersebut pada Docker Registry yang tersedia, contohnya yaitu Docker Hub.

1.2.4 Docker Container

Docker Container adalah instansi yang berjalan dari Docker Image. Docker Container adalah unit terisolasi yang dapat berkomunikasi dengan sistem operasi host, tetapi tetap terpisah dari container lainnya. Mereka menyediakan lingkungan yang konsisten dan dapat diulang, memungkinkan pengembang untuk menjalankan aplikasi tanpa khawatir tentang dependensi atau konfigurasi sistem operasi host. Container dapat dijalankan, dihentikan, dihapus, dan didistribusikan dengan cepat dan mudah.

1.2.5 Dockerfile

Dockerfile merupakan file teks biasa yang berisi serangkaian instruksi yang bisa kita gunakan untuk membuat sebuah Docker Image. Anggap saja semua instruksi untuk menjalankan aplikasi kita, kita simpan di dalam Dockerfile, nanti Dockerfile tersebut akan dieksekusi sebagai perintah untuk membuat Docker Image. Untuk membuat Docker Image dari Dockerfile, kita bisa menggunakan perintah **docker build**.

Ada banyak tipe instruksi yang bisa digunakan dalam Dockerfile, untuk lebih jelasnya kita bisa akses [di sini](#). Tapi di sini kita hanya membahas beberapa saja yang penting, seperti berikut:

INSTRUKSI	DESKRIPSI
FROM	Instruksi FROM memberi tahu Docker basis image apa yang akan digunakan.
RUN	RUN sebuah instruksi untuk mengeksekusi perintah di dalam image pada saat build stage. Biasanya digunakan untuk menginstal software dan menjalankan script, command, dan tugas-tugas lainnya.
CMD	CMD atau Command, merupakan instruksi yang digunakan ketika Docker Container dijalankan
ADD/COPY	ADD dan COPY adalah instruksi yang dapat digunakan untuk menambahkan file dari source ke dalam folder destination di Docker Image

USER	USER adalah instruksi yang digunakan untuk mengubah user atau user group ketika Docker Image dijalankan
WORKDIR	WORKDIR adalah instruksi untuk menentukan direktori / folder untuk menjalankan instruksi RUN, CMD, ENTRYPOINT, COPY dan ADD
LABEL	Instruksi LABEL merupakan instruksi yang digunakan untuk menambahkan metadata ke dalam Docker Image yang kita buat

BAB II

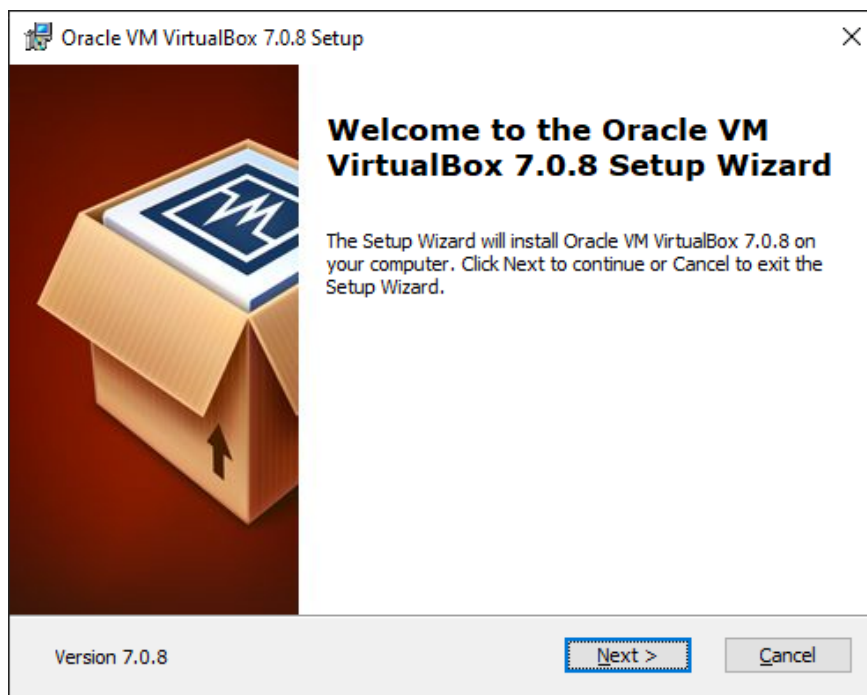
Persiapan LAB

Minimum spesifikasi

1. Windows 10 64bit
2. RAM 4GB
3. Storage tersisa 10GB
4. VirtualBox 7.0.0+

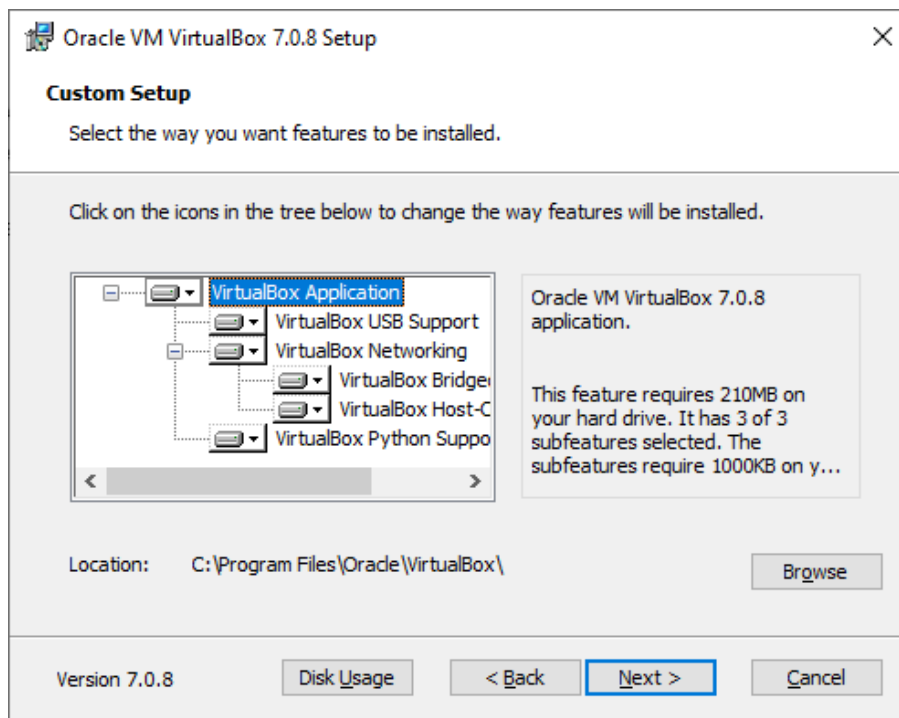
Install VirtualBox

1. Jalankan file installer VirtualBox



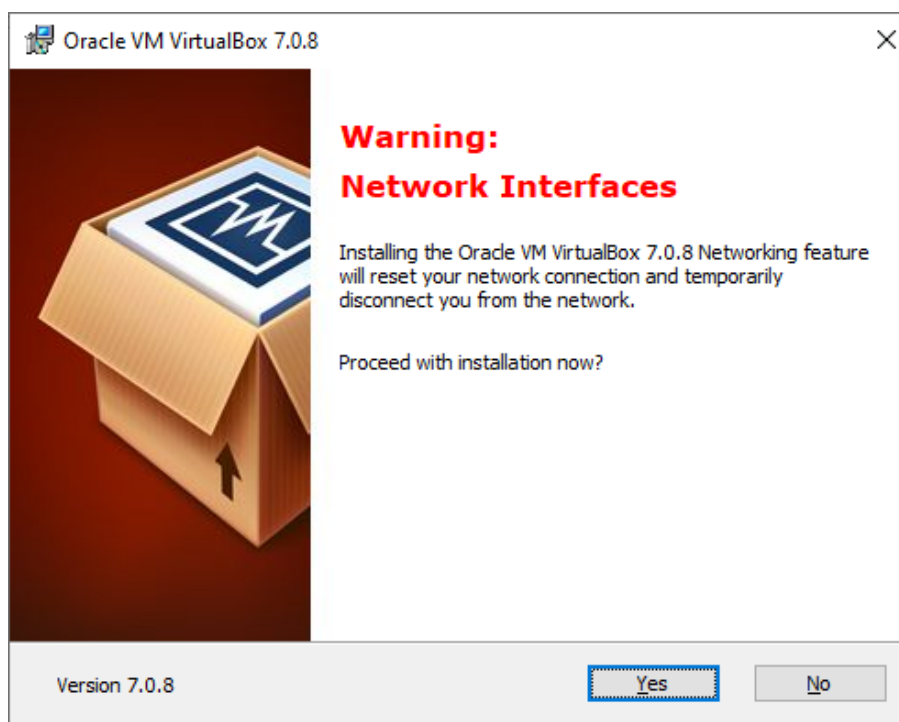
Klik next

2. Selanjutnya akan muncul menu fitur apa yang akan diinstall



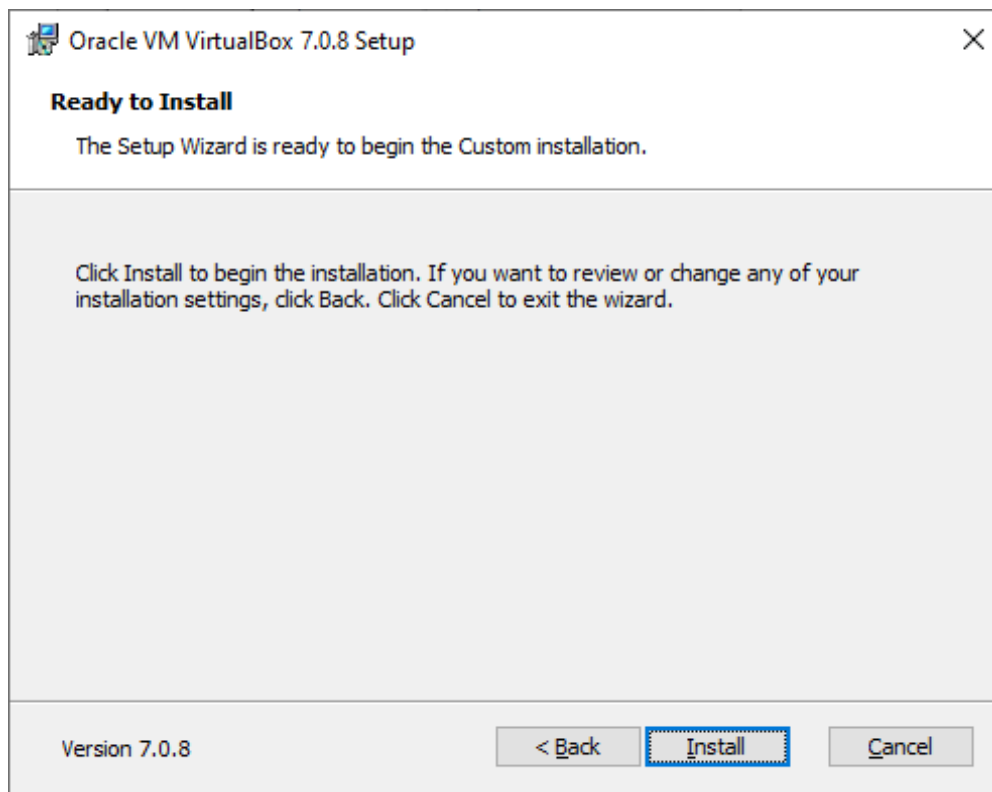
Untuk melanjutkan, klik next

3. Kemudian akan muncul menu untuk menginstall network interface

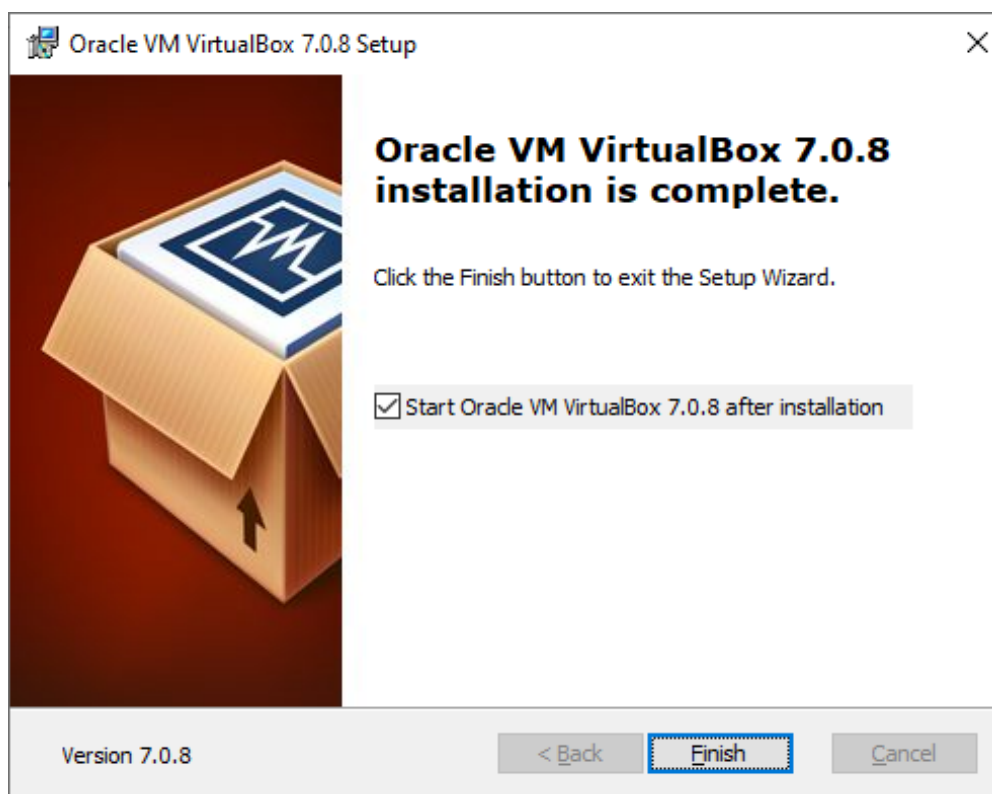


Klik "Yes"

4. Klik install untuk memulai proses instalasi

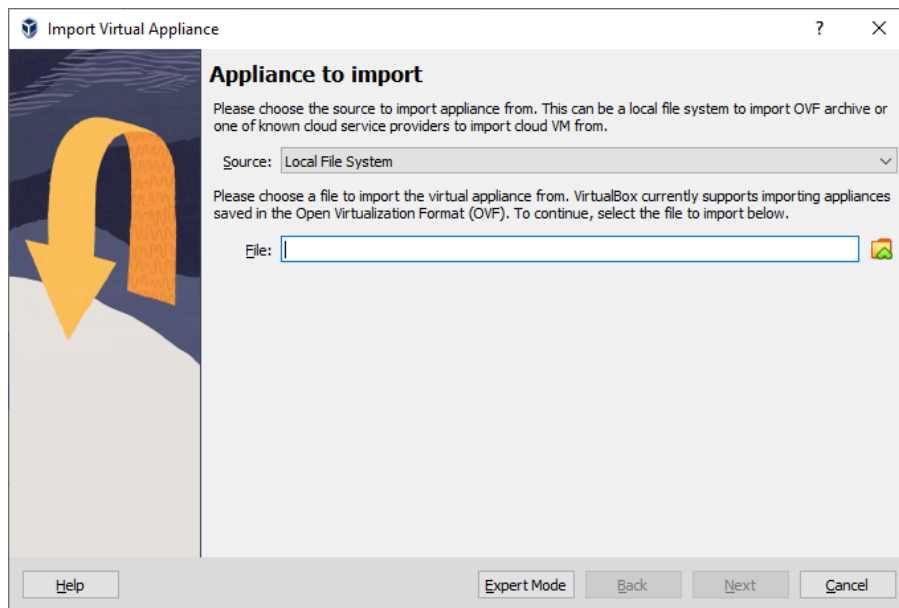


5. Instalasi selesai.

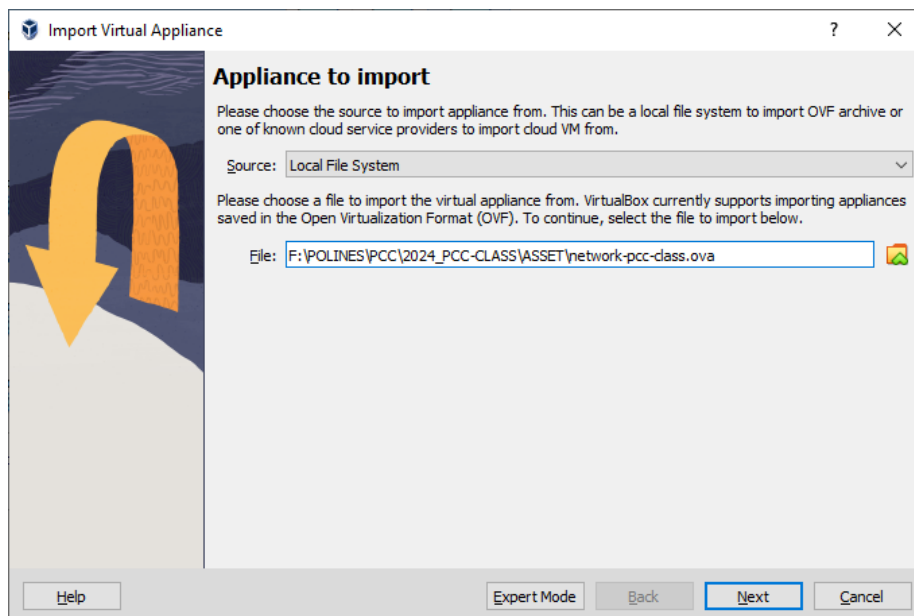


Import Lab:

1. Buka VirtualBox
2. Klik File > Import Appliance

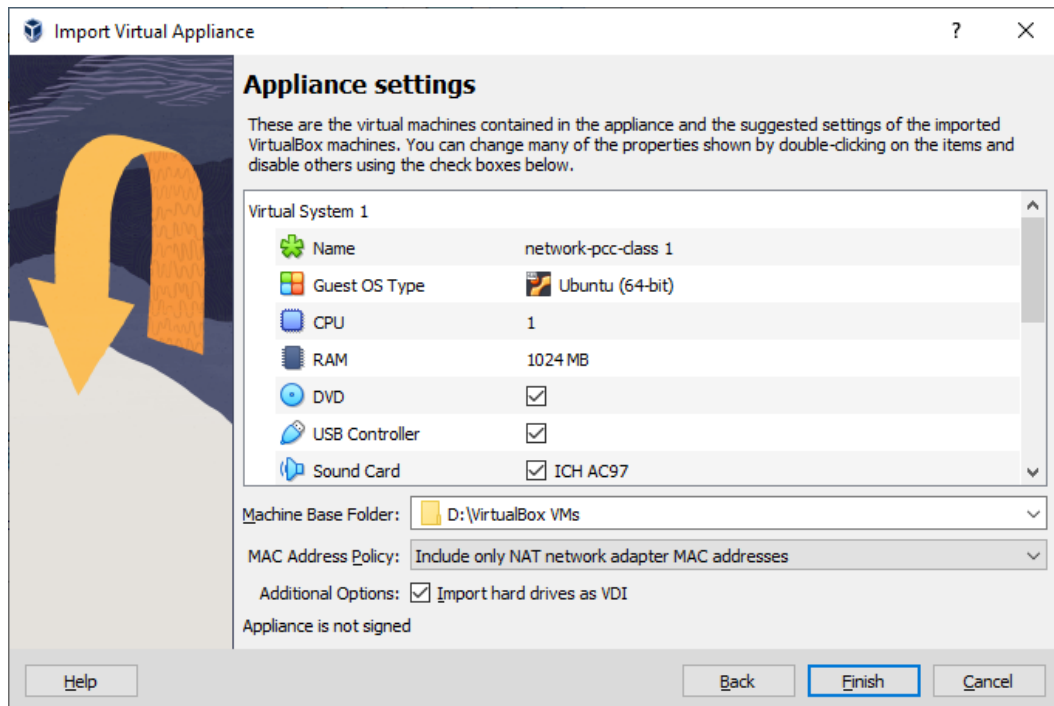


3. Kemudian pilih file .ova yang sudah di download sebelumnya.



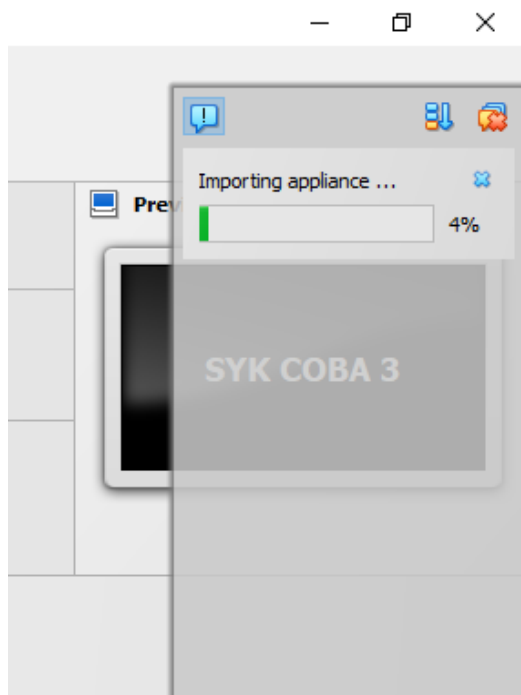
Klik Next

4. Kemudian akan muncul informasi dan pengaturan tentang VM yang akan di-import. Kita dapat mengubah beberapa opsi pengaturan tersebut, contohnya kita dapat mengubah nilai alokasi ram untuk Virtual Machine kita.



Jika dirasa tidak ada yang ingin diubah, bisa langsung klik Finish.

5. Kemudian akan muncul notifikasi yang berisi progress proses di sebelah kanan window.



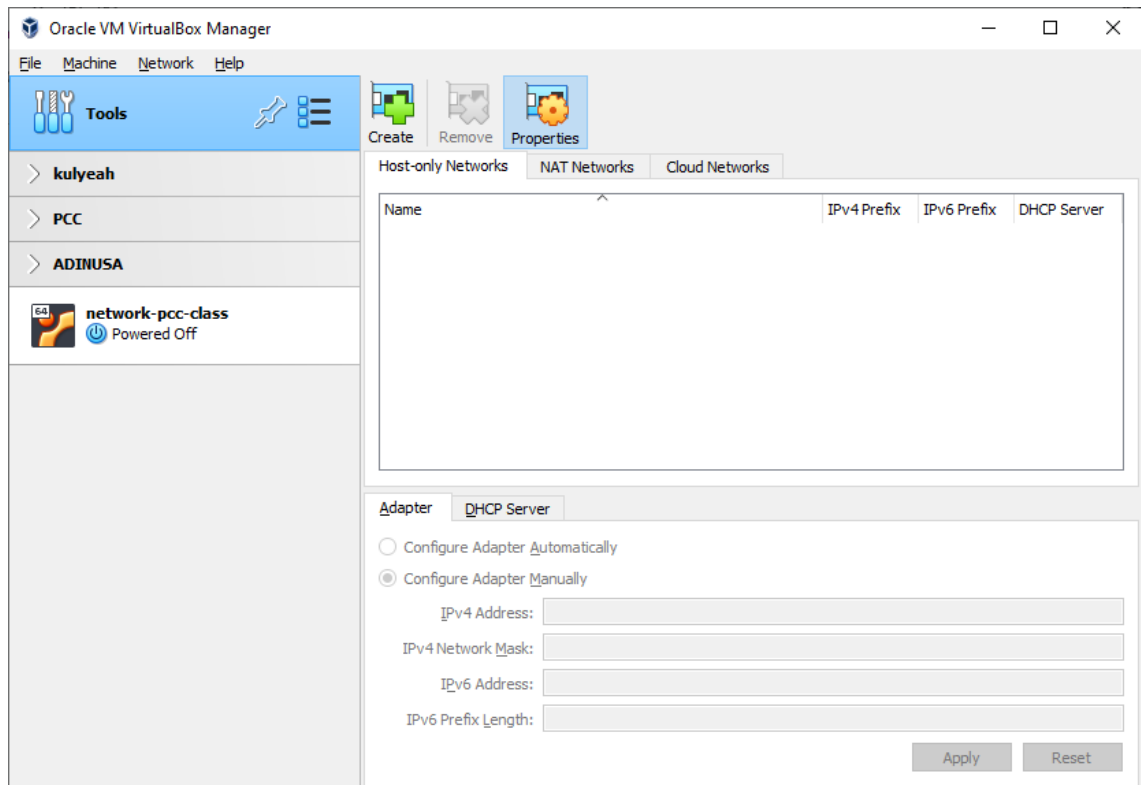
Tunggu hingga proses import selesai.

6. Proses import selesai.

Setup Network

Disini kita akan mengatur network interface yang nantinya akan kita gunakan untuk akses internet dan local VM Ubuntu kita.

1. Masuk ke menu 'Tools', kemudian klik tab 'Host-only Network'



2. Klik Create. Kemudian tunggu sampai proses membuat network interface selesai.
3. Jika pembuatan network interface sudah selesai, maka akan muncul Host-only interface baru beserta dengan IPnya

Create Remove Properties

Host-only Networks NAT Networks Cloud Networks

Name	IPv4 Prefix	IPv6 Prefix	DHCP Server
VirtualBox Host-Only Ethernet Adapter	192.168.29.1/24		Disabled

Adapter DHCP Server

☐ Configure Adapter Automatically

☒ Configure Adapter Manually

IPv4 Address: 192.168.29.1

IPv4 Network Mask: 255.255.255.0

IPv6 Address: fe80::424f:3c70:3398:300a

IPv6 Prefix Length: 64

Apply Reset

4. Lalu isi dengan IP sebagai berikut:

192.168.1.1

Adapter DHCP Server

☐ Configure Adapter Automatically

☒ Configure Adapter Manually

IPv4 Address: 192.168.1.1

IPv4 Network Mask: 255.255.255.0

IPv6 Address: fe80::424f:3c70:3398:300a

IPv6 Prefix Length: 64

Apply Reset

Kemudian klik 'apply'

5. Pindah ke tab 'NAT Network', kemudian klik 'Create' untuk membuat interface NAT baru

Create

Remove

Properties

Host-only Networks

NAT Networks

Cloud Networks

Name	IPv4 Prefix	IPv6 Prefix	DHCP Server
NatNetwork	10.0.2.0/24		Enabled

General Options

Port Forwarding

Name:

NatNetwork

IPv4 Prefix:

10.0.2.0/24

☒ Enable DHCP

☐ Enable IPv6

IPv6 Prefix:

☐ Advertise Default IPv6 Route

Apply

Reset

6. Ubah IP sebagai berikut:
192.168.0.0/24

General Options

Port Forwarding

Name:

NatNetwork

IPv4 Prefix:

192.168.0.0/24

☒ Enable DHCP

☐ Enable IPv6

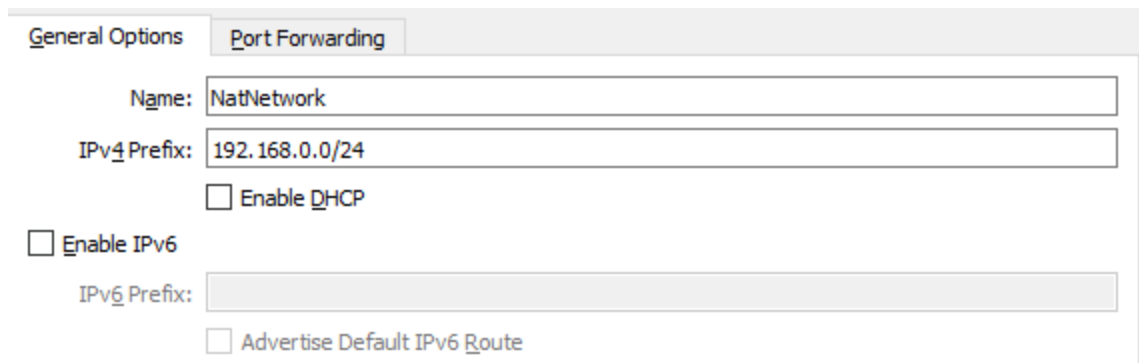
IPv6 Prefix:

☐ Advertise Default IPv6 Route

Apply

Reset

7. Uncheck Enable DHCP. Kemudian klik apply



General Options Port Forwarding

Name: NatNetwork

IPv4 Prefix: 192.168.0.0/24

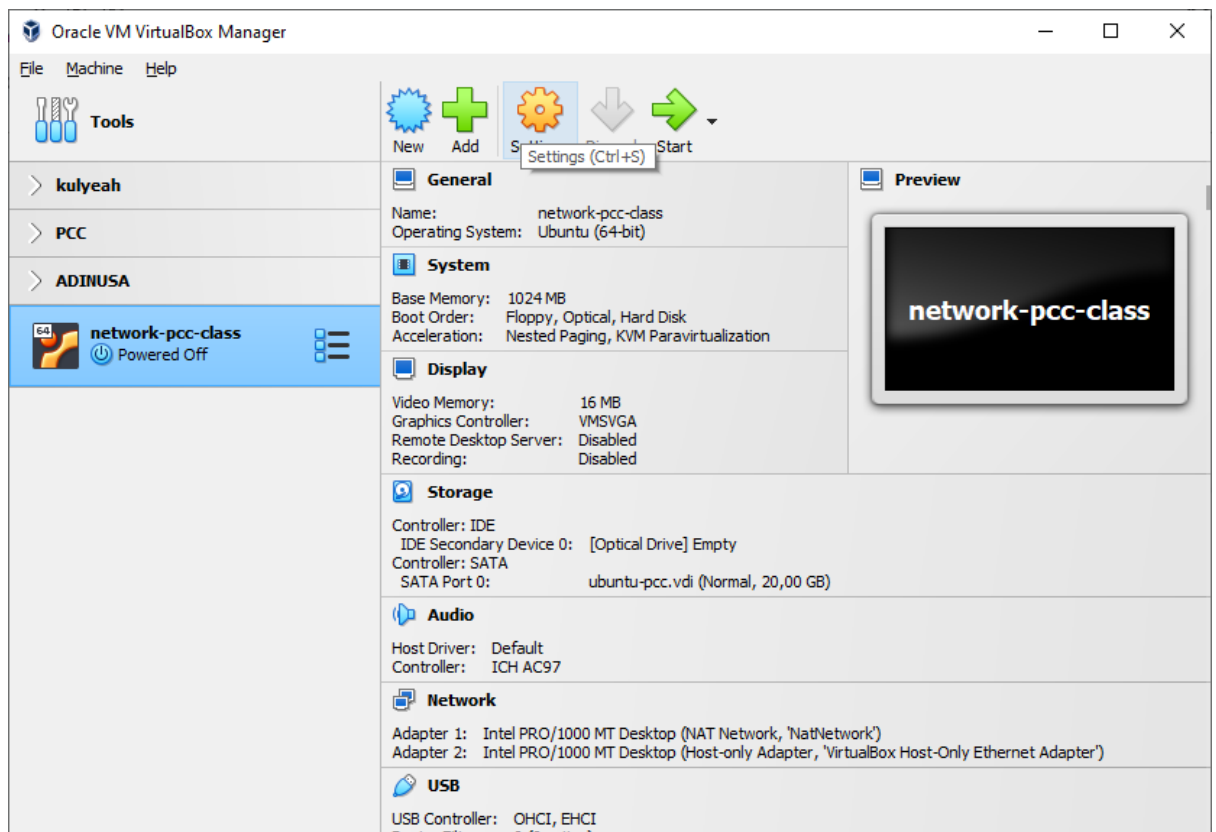
☐ Enable DHCP

☐ Enable IPv6

IPv6 Prefix:

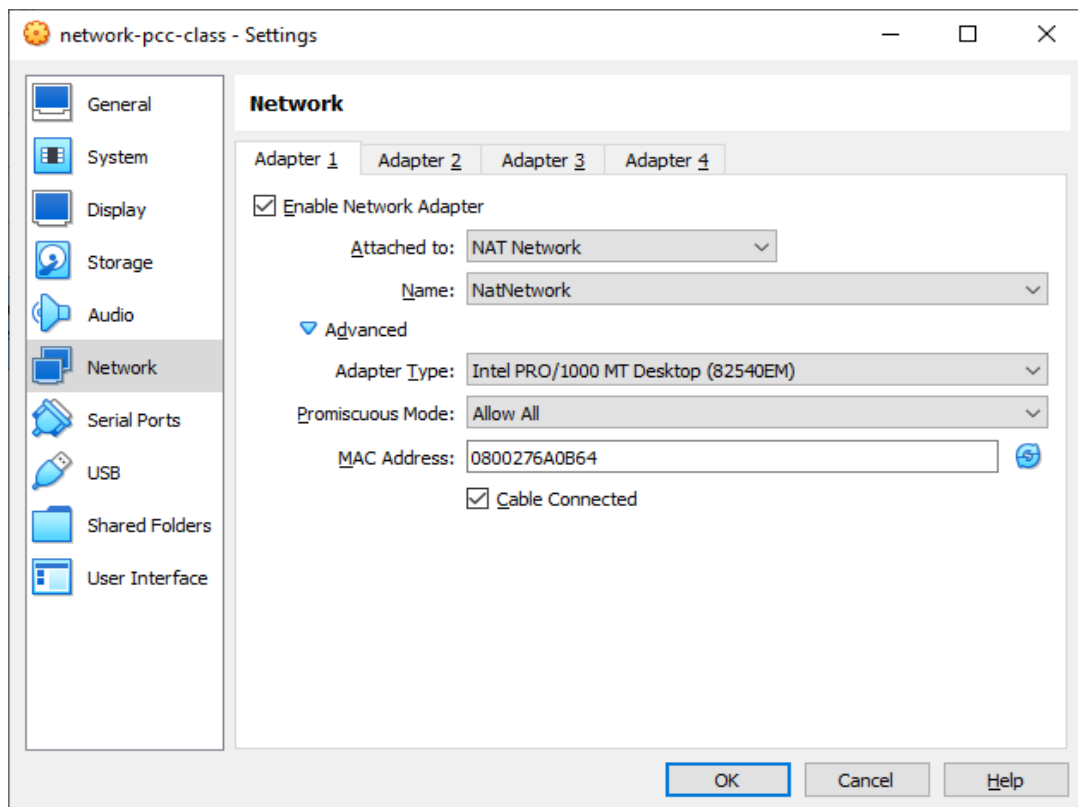
☐ Advertise Default IPv6 Route

8. Klik VM, kemudian masuk ke settings

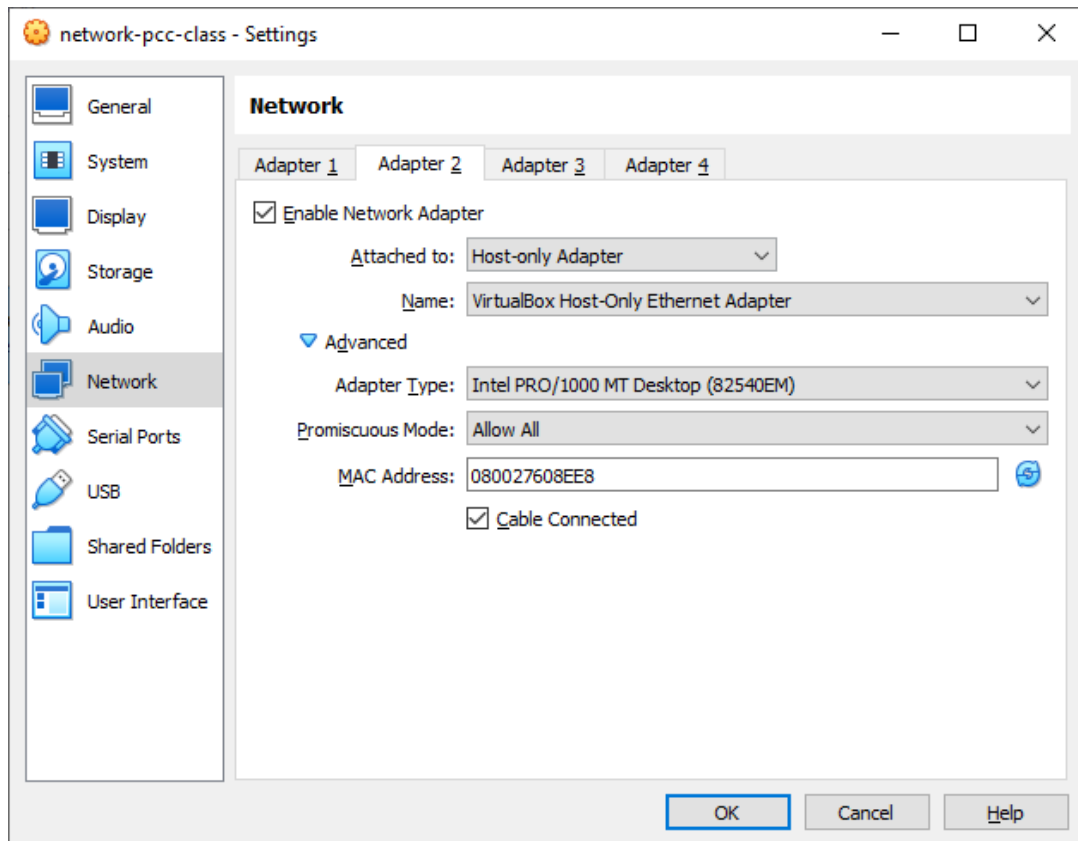


9. Masuk ke tab 'Network'.

10. Pada 'Adapter 1', pilih Nat Network.

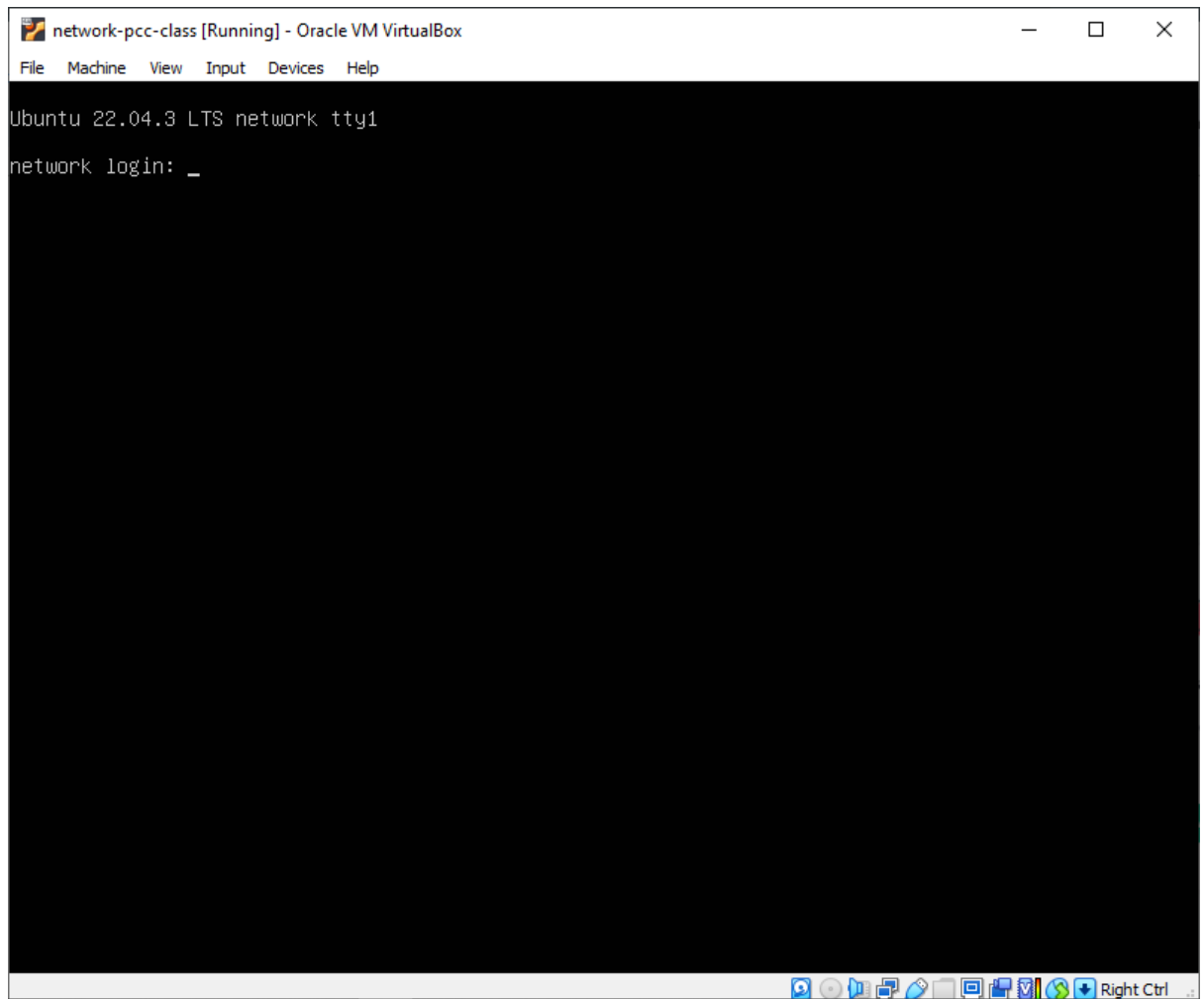


11. Lalu pada 'Adapter 2', pilih Host-only Adapter



12. Selanjutnya kita akan konfigurasi IP untuk linuxnya.

13. Hidupkan VM



14. Login dengan username: **pcc-class** dan password: **network2024**

15. Kemudian masuk ke direktori netplan dimana kita akan mengkonfigurasi network Ubuntu disana. Ketikkan perintah **cd /etc/netplan**

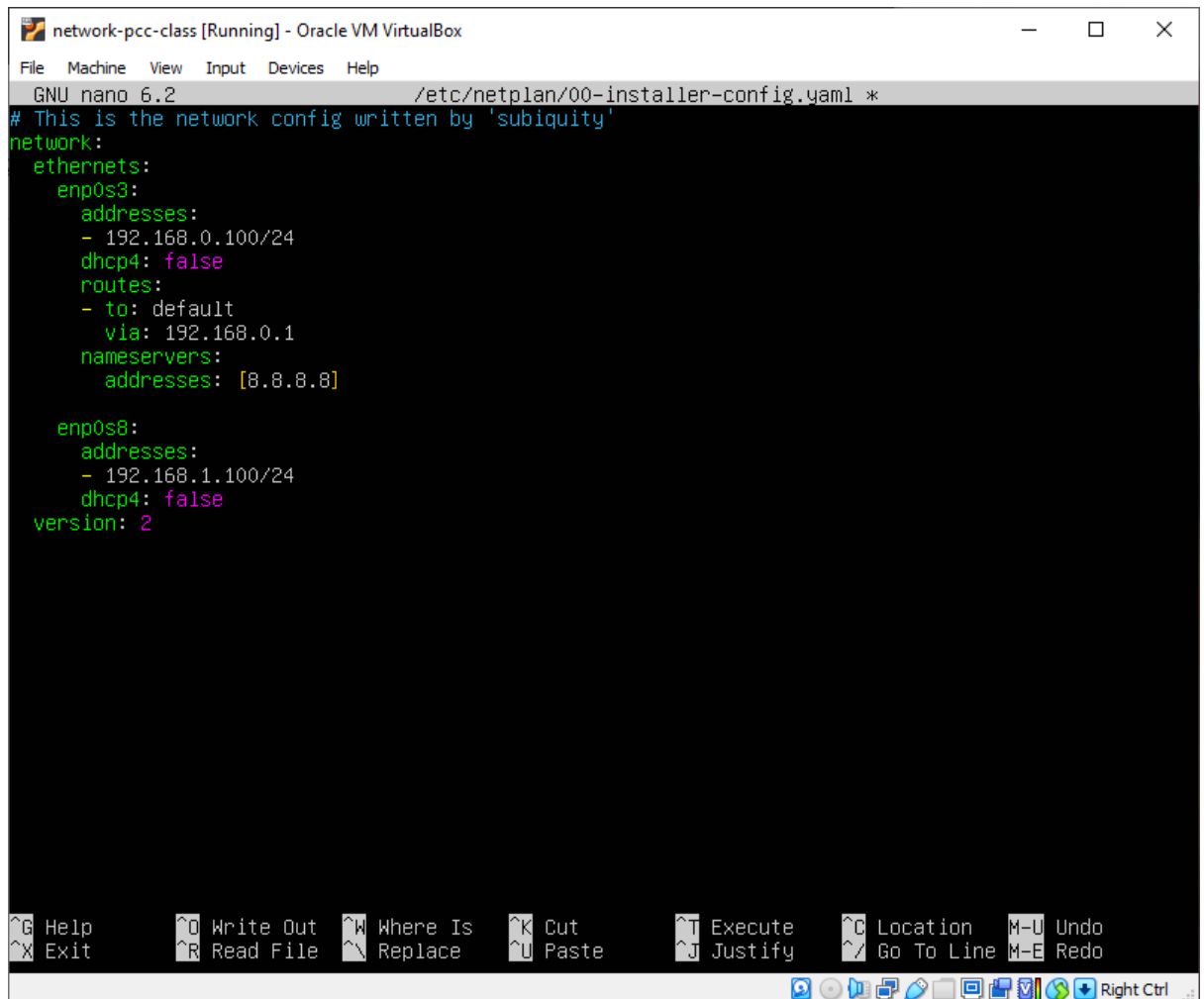
```
pcc-class@network:~$ cd /etc/netplan/  
pcc-class@network:/etc/netplan$ ls  
00-installer-config.yaml  
pcc-class@network:/etc/netplan$ _
```

Direktori **/etc/netplan** digunakan untuk mengkonfigurasi jaringan pada VM Ubuntu versi 22 yang kita gunakan sekarang ini.

16. Kita edit file **00-installer-config.yaml** dengan mengetikkan perintah

sudo nano 00-installer-config.yaml

17. Kemudian edit seperti berikut:



The screenshot shows a terminal window titled "network-pcc-class [Running] - Oracle VM VirtualBox". The terminal is running GNU nano 6.2 and editing the file /etc/netplan/00-installer-config.yaml. The content of the file is a network configuration for two interfaces, enp0s3 and enp0s8. The configuration includes IP addresses, DHCP settings, routes, and nameservers. The terminal also shows a menu bar with various editing and execution commands.

```
network-pcc-class [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 6.2 /etc/netplan/00-installer-config.yaml *
# This is the network config written by 'subiquity'
network:
  ethernets:
    enp0s3:
      addresses:
        - 192.168.0.100/24
      dhcp4: false
      routes:
        - to: default
          via: 192.168.0.1
      nameservers:
        addresses: [8.8.8.8]

    enp0s8:
      addresses:
        - 192.168.1.100/24
      dhcp4: false
  version: 2

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location  M-U Undo
^X Exit      ^R Read File ^P Replace   ^U Paste     ^J Justify   ^_ Go To Line M-E Redo
                                                    Right Ctrl
```

```
network:
  ethernets:
    enp0s3:
      addresses:
        - 192.168.0.100/24
      dhcp4: false
      routes:
        - to: default
          via: 192.168.0.1
      nameservers:
        addresses: [8.8.8.8]

    enp0s8:
      addresses:
        - 192.168.1.100/24
      dhcp4: false
  version: 2
```

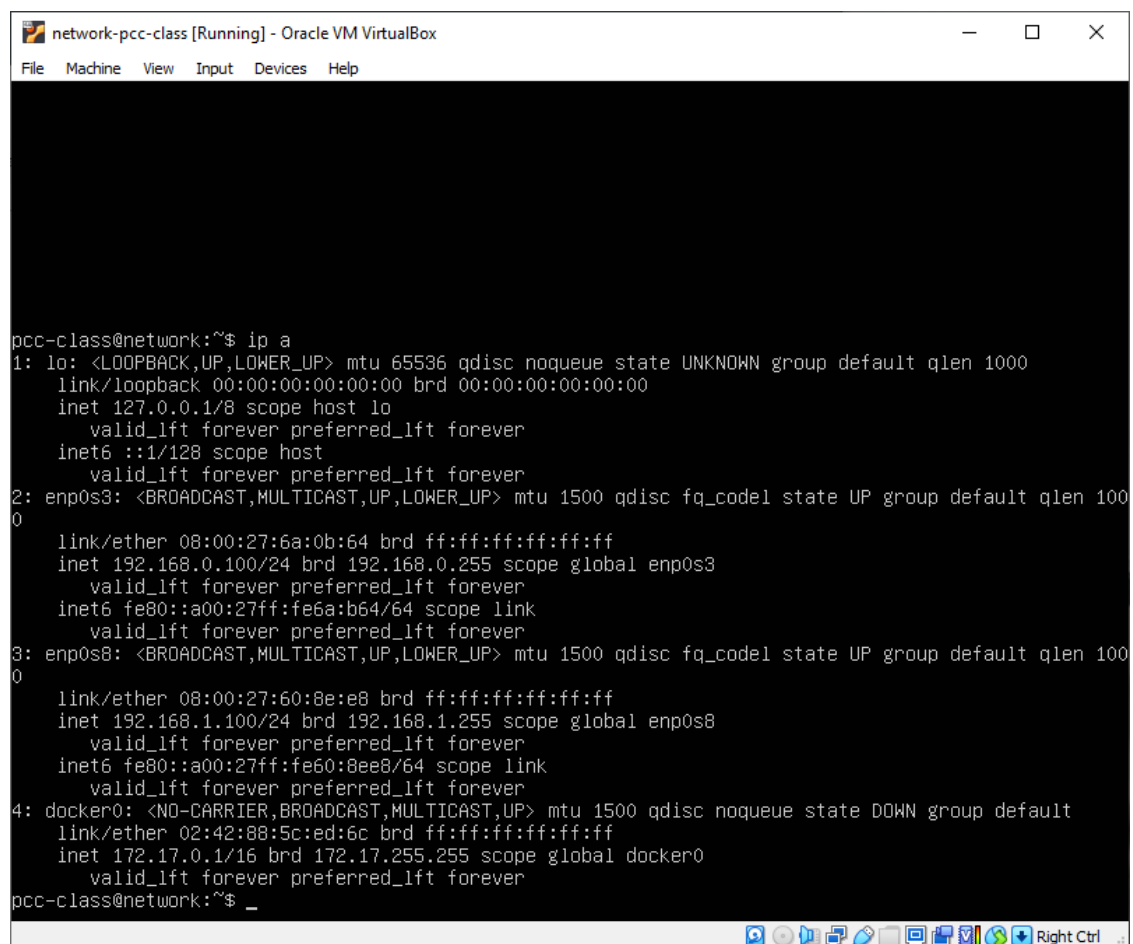
Dapat dilihat, kita akan mengkonfigurasi 2 network interface, yakni **enp0s3** dan **enp0s8** dimana masing-masing interface menghubungkan 2 adapter NAT dan Host-only sesuai dengan adapter yang kita atur untuk VM Ubuntu kita.

- **enp0s3** → Adapter 1 (NAT Network)
- **enp0s8** → Adapter 2 (Host-only Network)

18. Simpan dengan mengetikkan **ctrl+X, Y, Enter**

19. Gunakan perintah **sudo netplan apply** untuk menerapkan konfigurasi jaringan terbaru

20. Lalu Verifikasi dengan perintah **ip a**.



```
pcc-class@network:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:6a:0b:64 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.100/24 brd 192.168.0.255 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe6a:b64/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:60:8e:e8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.100/24 brd 192.168.1.255 scope global enp0s8
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe60:8ee8/64 scope link
        valid_lft forever preferred_lft forever
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:88:5c:ed:6c brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
pcc-class@network:~$
```

21. Kemudian verifikasi dengan ping ke internet

ping google.com

```
network-pcc-class [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

pcc-class@network:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:6a:0b:64 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.100/24 brd 192.168.0.255 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe6a:b64/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:60:8e:e8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.100/24 brd 192.168.1.255 scope global enp0s8
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe60:8ee8/64 scope link
        valid_lft forever preferred_lft forever
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:88:5c:ed:6c brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
pcc-class@network:~$ ping google.com -c 4
PING google.com (172.217.194.113) 56(84) bytes of data:
64 bytes from si-in-f113.1e100.net (172.217.194.113): icmp_seq=1 ttl=103 time=42.2 ms
64 bytes from si-in-f113.1e100.net (172.217.194.113): icmp_seq=2 ttl=103 time=28.4 ms
64 bytes from si-in-f113.1e100.net (172.217.194.113): icmp_seq=3 ttl=103 time=33.0 ms
64 bytes from si-in-f113.1e100.net (172.217.194.113): icmp_seq=4 ttl=103 time=34.1 ms

--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 28.408/34.445/42.202/4.967 ms
pcc-class@network:~$ _
```

Instalasi Tools

Dalam praktek selanjutnya, kita akan banyak copy-paste script. Dikarenakan dalam Ubuntu CLI tidak bisa menggunakan copy-paste, maka kita akan menggunakan alternatif lain, yaitu remote SSH melalui Terminal atau Command Prompt Windows.

1. Buka aplikasi Terminal atau Command Prompt pada windows.
2. Kemudian ketikkan `ssh pcc-class@192.168.1.100`

```
PS C:\Users\FATUR> ssh pcc-class@192.168.1.100
The authenticity of host '192.168.1.100 (192.168.1.100)' can't be established.
ECDSA key fingerprint is SHA256:mDvcdKWw6WspmeILplg6MPWmPGIX/EGMDMesR+ewDgQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes|
```

Jika muncul dialog untuk meminta RSA-Key, ketik **yes**

```
PS C:\Users\FATUR> ssh pcc-class@192.168.1.100
The authenticity of host '192.168.1.100 (192.168.1.100)' can't be established.
ECDSA key fingerprint is SHA256:mDvcdKWw6WspmeILplg6MPWmPGIX/EGMDMesR+ewDgQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.100' (ECDSA) to the list of known hosts.
pcc-class@192.168.1.100's password: |
```

3. Kemudian Masukkan password **network2024**

```
PS C:\Users\FATUR> ssh pcc-class@192.168.1.100
The authenticity of host '192.168.1.100 (192.168.1.100)' can't be established.
ECDSA key fingerprint is SHA256:mDvcdKWw6WspmeILplg6MPWmPGIX/EGMDMesR+ewDgQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.100' (ECDSA) to the list of known hosts.
pcc-class@192.168.1.100's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-94-generic x86_64)

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

System information as of Fri Feb 23 03:18:11 AM UTC 2024

System load:  0.19921875      Users logged in:      1
Usage of /:   64.2% of 9.75GB IPv4 address for docker0: 172.17.0.1
Memory usage: 26%           IPv4 address for enp0s3: 192.168.0.100
Swap usage:   0%             IPv4 address for enp0s8: 192.168.1.100
Processes:   107

Expanded Security Maintenance for Applications is not enabled.

55 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Fri Feb 23 02:39:25 2024
pcc-class@network:~$
```

Login telah sukses

4. Selanjutnya kita akan meginstall docker. Jalankan perintah berikut:

```

sudo apt-get update
sudo apt-get install ca-certificates curl gnupg lsb-release -y
sudo mkdir -m 0755 -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin -y

```

Paste dan jalankan kode tersebut pada Terminal

5. Jika instalasi telah selesai, verifikasi dengan menjalankan perintah

docker version

```

pcc-class@network:~$ docker version
Client: Docker Engine - Community
Version:      25.0.3
API version:  1.44
Go version:   go1.21.6
Git commit:   4debf41
Built:        Tue Feb  6 21:13:09 2024
OS/Arch:      linux/amd64
Context:      default

Server: Docker Engine - Community
Engine:
Version:      25.0.3
API version:  1.44 (minimum version 1.24)
Go version:   go1.21.6
Git commit:   f417435
Built:        Tue Feb  6 21:13:09 2024
OS/Arch:      linux/amd64
Experimental: false
containerd:
Version:      1.6.28
GitCommit:    ae07eda36dd25f8a1b98dfbf587313b99c0190bb
runc:
Version:      1.1.12
GitCommit:    v1.1.12-0-g51d5e94
docker-init:
Version:      0.19.0
GitCommit:    de40ad0

```

6. Tambahkan user ke Docker Group

```
groupadd docker  
sudo usermod -aG docker $USER  
sudo chmod 666 /var/run/docker.sock
```


BAB III

Langkah-langkah

Manajemen Docker Image

1. Untuk melihat image yang ada, jalankan perintah

docker image ls

```
pcc-class@network:~/coba$ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
phpmyadmin	latest	a9695a48170e	11 days ago	562MB
nginx	latest	e4720093a3c1	2 weeks ago	187MB
registry.adinusa.id/btacademy/mysql	latest	8189e588b0e8	10 months ago	564MB
redis	3	87856cc39862	5 years ago	76MB

2. Untuk mengambil image dari registry, dapat gunakan perintah

docker pull <nama-image>

```
pcc-class@network:~/coba$ docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
4abcf2066143: Pull complete
Digest: sha256:c5b1261d6d3e43071626931fc004f70149baeba2c8ec672bd4f27761f8e1ad6b
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
pcc-class@network:~/coba$ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
phpmyadmin	latest	a9695a48170e	11 days ago	562MB
nginx	latest	e4720093a3c1	2 weeks ago	187MB
alpine	latest	05455a08881e	4 weeks ago	7.38MB
registry.adinusa.id/btacademy/mysql	latest	8189e588b0e8	10 months ago	564MB
redis	3	87856cc39862	5 years ago	76MB

3. Untuk menghapus image, gunakan perintah

docker image rm <nama/image-id>

```
pcc-class@network:~/coba$ docker image rm alpine
Untagged: alpine:latest
Untagged: alpine@sha256:c5b1261d6d3e43071626931fc004f70149baeba2c8ec672bd4f27761f8e1ad6b
Deleted: sha256:05455a08881ea9cf0e752bc48e61bbd71a34c029bb13df01e40e3e70e0d007bd
Deleted: sha256:d4fc045c9e3a848011de66f34b81f052d4f2c15a17bb196d637e526349601820
pcc-class@network:~/coba$ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
phpmyadmin	latest	a9695a48170e	11 days ago	562MB
nginx	latest	e4720093a3c1	2 weeks ago	187MB
registry.adinusa.id/btacademy/mysql	latest	8189e588b0e8	10 months ago	564MB
redis	3	87856cc39862	5 years ago	76MB

4. Untuk mengambil image dengan versi/tag tertentu, gunakan perintah

docker image pull <nama-image:tag>

```
pcc-class@network:~/latihan2$ docker pull redis:6.0.20
6.0.20: Pulling from library/redis
elcaac4eb9d2: Already exists
ed7c5e39c4ce: Pull complete
2a829b3e35ab: Pull complete
71d94ae462e7: Pull complete
e6a02c0dce36: Pull complete
3d113e0f7953: Pull complete
4f4fb700ef54: Pull complete
6108f9e6fb0c: Pull complete
Digest: sha256:4cd0fb840eddd607bd7a3b3ca91a950122c2e2faddb8145ad39dd4e4dfdd038d
Status: Downloaded newer image for redis:6.0.20
docker.io/library/redis:6.0.20
pcc-class@network:~/latihan2$ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
phpmyadmin	latest	a9695a48170e	12 days ago	562MB
nginx	latest	e4720093a3c1	2 weeks ago	187MB
redis	6.0.20	5e50eed779b1	2 months ago	126MB
registry.adinusa.id/btacademy/mysql	latest	8189e588b0e8	10 months ago	564MB
redis	3	87856cc39862	5 years ago	76MB

Latihan 1 (docker image)

1. pull image alpine dengan versi 3.16 (bisa lihat di hub.docker.com)

Manajemen container

1. Untuk melihat container yang ada, gunakan perintah

docker container ls

```
pcc-class@network:~/coba$ docker container ls
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
pcc-class@network:~/coba$ docker container ls -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
```

Karena kita belum menjalankan container, maka pada saat kita list container, outputnya masih kosong.

Kita juga dapat menjalankan perintah **docker ps** untuk melihat container yang ada

```
pcc-class@network:~/coba$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
pcc-class@network:~/coba$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
pcc-class@network:~/coba$ |
```

2. Untuk menjalankan container, gunakan perintah

docker run -d <nama-image>

contoh: **docker run -d nginx**

```
pcc-class@network:~/coba$ docker run -d nginx
40201d37aa6117c49f306fe97c6546d1782c56852c670ba23479aeaf053dab5
pcc-class@network:~/coba$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED       STATUS       PORTS       NAMES
40201d37aa61   nginx     "/docker-entrypoint..." 3 seconds ago Up 1 second  80/tcp      gracious_sanderson
pcc-class@network:~/coba$ |
```

Container dari image nginx telah berjalan. Dapat dilihat disana terdapat container yang berjalan dari image nginx. Nama dari container akan otomatis diberikan oleh docker jika kita tidak memberi nama pada saat membuat container.

3. Untuk memberi nama container, gunakan perintah

docker run -d --name <nama-container> <image>

contoh: **docker run -d --name nginx-container nginx**

```
pcc-class@network:~/coba$ docker run -d --name nginx-container nginx
b5c3eaf165d45ed7c6bc69d333b2379f92b778a069da14222e3c45a4ec8e2b1
pcc-class@network:~/coba$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
b5c3eaf165d4	nginx	"/docker-entrypoint..."	27 seconds ago	Up 26 seconds	80/tcp	nginx-container
40201d37aa61	nginx	"/docker-entrypoint..."	5 minutes ago	Up 5 minutes	80/tcp	gracious_sanderson

```
pcc-class@network:~/coba$
```

4. Menghentikan container yang sedang berjalan

docker container stop <nama/id-container>

contoh: **docker container stop nginx-container**

```
pcc-class@network:~/coba$ docker container stop nginx-container
nginx-container
pcc-class@network:~/coba$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
40201d37aa61	nginx	"/docker-entrypoint..."	9 minutes ago	Up 9 minutes	80/tcp	gracious_sanderson

```
pcc-class@network:~/coba$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
b5c3eaf165d4	nginx	"/docker-entrypoint..."	4 minutes ago	Exited (0) 5 seconds ago		nginx-container
40201d37aa61	nginx	"/docker-entrypoint..."	9 minutes ago	Up 9 minutes	80/tcp	gracious_sanderson

```
pcc-class@network:~/coba$
```

Dapat dilihat status pada container my-container berubah menjadi "Exited 5 seconds ago".

5. Menghidupkan container

docker container start <nama/id-container>

contoh: **docker container start nginx-container**

```
pcc-class@network:~/coba$ docker container start nginx-container
nginx-container
pcc-class@network:~/coba$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
b5c3eaf165d4	nginx	"/docker-entrypoint..."	46 minutes ago	Up 2 seconds	80/tcp	nginx-container
40201d37aa61	nginx	"/docker-entrypoint..."	50 minutes ago	Up 50 minutes	80/tcp	gracious_sanderson

```
pcc-class@network:~/coba$
```

Dapat dilihat status container sudah berjalan

6. Menghapus container

docker container rm <nama/id-container>

```
pcc-class@network:~/coba$ docker container stop gracious_sanderson
gracious_sanderson
pcc-class@network:~/coba$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
b5c3eaf165d4	nginx	"/docker-entrypoint..."	51 minutes ago	Up 4 minutes	80/tcp	nginx-container
40201d37aa61	nginx	"/docker-entrypoint..."	56 minutes ago	Exited (0) 4 seconds ago		gracious_sanderson

```
pcc-class@network:~/coba$ docker container rm gracious_sanderson
gracious_sanderson
pcc-class@network:~/coba$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
b5c3eaf165d4	nginx	"/docker-entrypoint..."	51 minutes ago	Up 4 minutes	80/tcp	nginx-container

```
pcc-class@network:~/coba$
```

7. Kita juga dapat menjalankan docker di port tertentu

docker run -d --name <nama-container> -p <port> <image>

contoh: **docker run -d --name my-nginx -p 8000 nginx**

```
pcc-class@network:~/coba$ docker run -d --name my-nginx -p 8000 nginx
7e38730db1e93b47ae3a713df5b5d14dfa85bcd2c8dc94161a89d8162955d811
pcc-class@network:~/coba$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
7e38730db1e9	nginx	"/docker-entrypoint..."	6 seconds ago	Up 4 seconds	80/tcp, 0.0.0.0:32768->8000/tcp,
:::32768->8000/tcp	my-nginx				

```
pcc-class@network:~/coba$
```

8. Kita dapat mem-publish (port-forwarding) aplikasi container kita

docker run -d -name <nama-container> -p <port-keluar>:<port-container> <image>

contoh: **docker run -d --name my-nginx -p 80:80 nginx**

artinya semua perangkat pada jaringan kita akan dapat mengakses container nginx pada port 80.

```
pcc-class@network:~/coba$ docker run -d --name my-nginx -p 80:80 nginx
bed7af4c47ceb31ee4350d07c0131ddc7a4c6de3a7c82fed10085a2d7dc70c38
pcc-class@network:~/coba$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
bed7af4c47ce	nginx	"/docker-entrypoint..."	3 seconds ago	Up 2 seconds	0.0.0.0:80->80/tcp,
:::80->80/tcp	my-nginx				

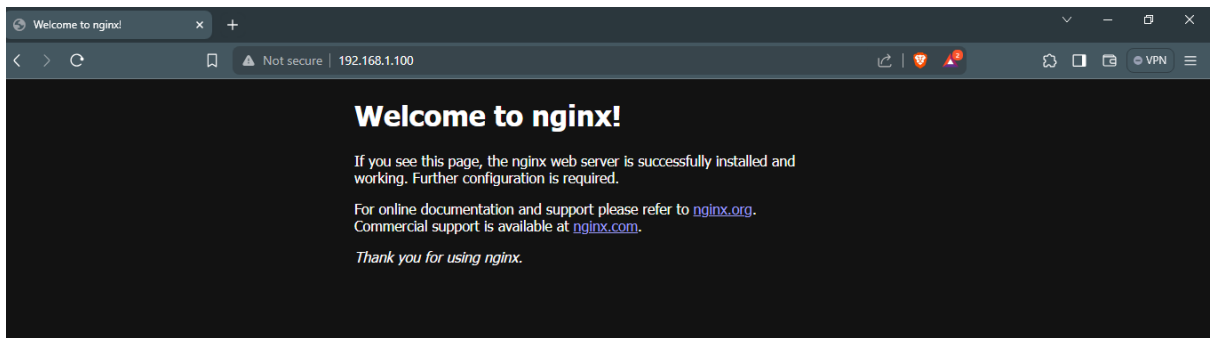
Kita coba cek dengan perintah curl apakah service nginx berjalan

```
pcc-class@network:~/coba$ curl localhost
<!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>
```

Kita coba cek juga menggunakan web browser pada laptop kita



9. Remote container

docker exec -it <nama/id-container> /bin/bash

contoh: **docker exec -it my-nginx /bin/bash**

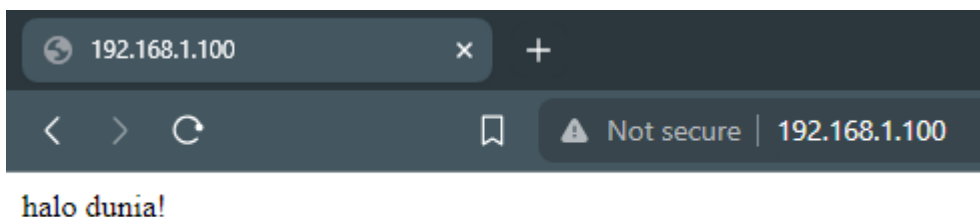
```
pcc-class@network:~/coba$ docker exec -it my-nginx /bin/bash
root@bed7af4c47ce:/# |
```

kita telah berhasil remote containernya

Kita coba untuk mengedit file tampilan nginx

```
root@bed7af4c47ce:/# echo "halo dunia!" > /usr/share/nginx/html/index.html
root@bed7af4c47ce:/# exit
exit
pcc-class@network:~/coba$ curl localhost
halo dunia!
pcc-class@network:~/coba$ |
```

coba cek di web browser



10. Environment Variable

Untuk memasukkan Environment Variable pada docker container kita, dapat menggunakan perintah **docker run -d -e <ENV=VALUE>**

contoh: **docker run -d --name my-mysql **
**-e MYSQL_ROOT_PASSWORD=supersecret **
**-e MYSQL_DATABASE=pcc -p 3306:3306 **
registry.adinusa.id/btacademy/mysql

```
pcc-class@network:~$ docker run -d --name my-mysql -e MYSQL_ROOT_PASSWORD=supersecret -e MYSQL_DATABASE=pcc
-p 3306:3306 registry.adinusa.id/btacademy/mysql
433ca50c3347c4d0389d285cd050321ae4d53f0086f503b3c33dec86876ebd6d
pcc-class@network:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
433ca50c3347	registry.adinusa.id/btacademy/mysql	"docker-entrypoint.s..."	11 seconds ago	Up 9 seconds
0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp		my-mysql		

11. Membuat link dengan container lain

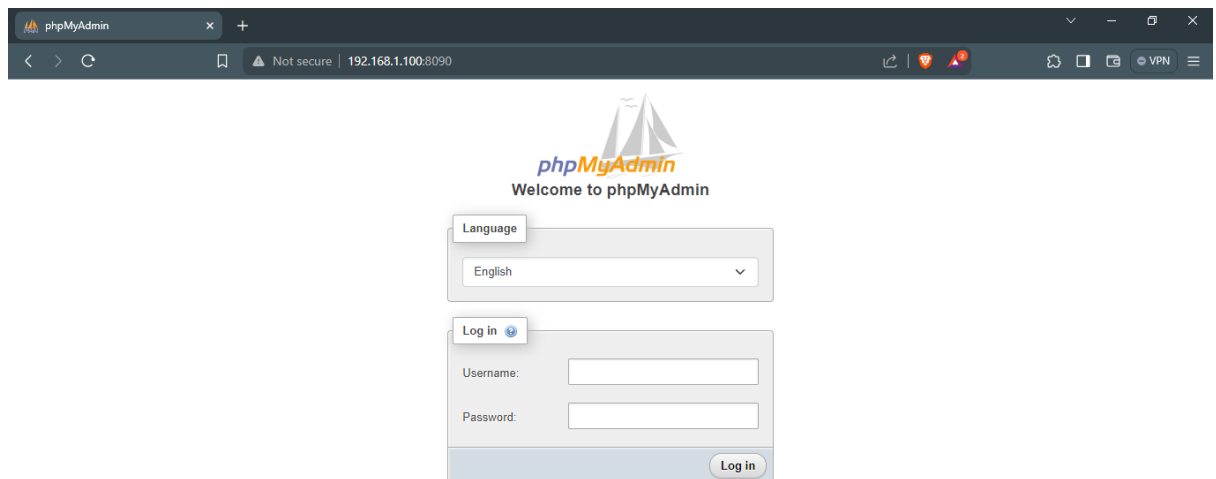
docker run -d --name <name> --link <link ke container> <image>

contoh: **docker run -d --name my-phpmyadmin **
--link my-mysql:db -p 8090:80 phpmyadmin

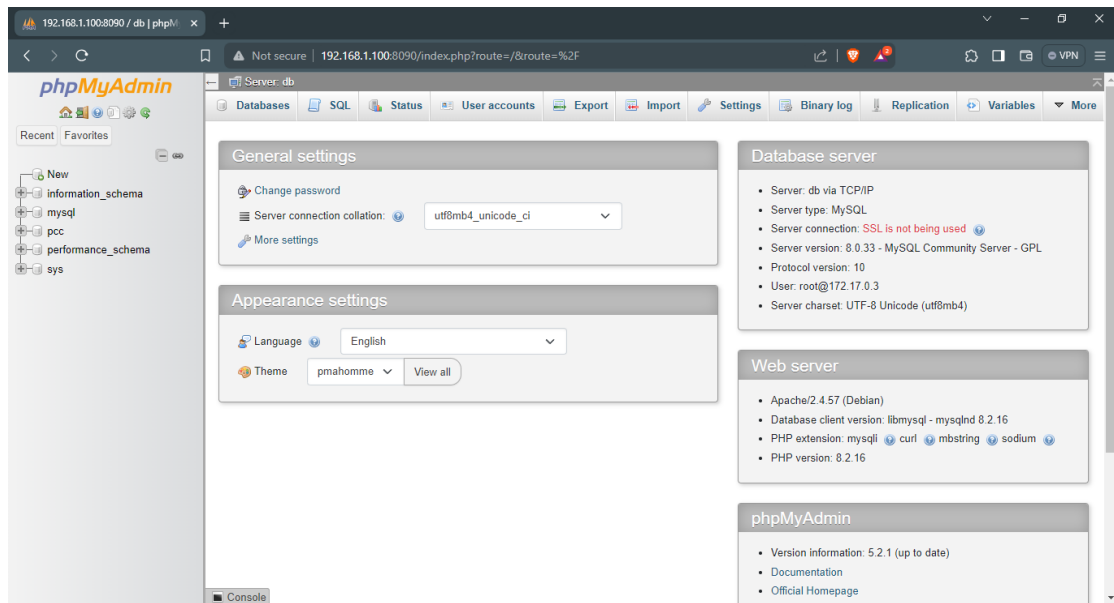
```
pcc-class@network:~$ docker run --name my-phpmyadmin -d --link my-mysql:db -p 8090:80 phpmyadmin
be563086027907a2aa344d1bdd56b9d762ee42078c9c7ef61245998024684a06
pcc-class@network:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
be5630860279	phpmyadmin	"/docker-entrypoint..."	26 seconds ago	Up 14 seconds
0.0.0.0:8090->80/tcp, :::8090->80/tcp		my-phpmyadmin		
433ca50c3347	registry.adinusa.id/btacademy/mysql	"docker-entrypoint.s..."	18 minutes ago	Up 18 minutes
0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp		my-mysql		

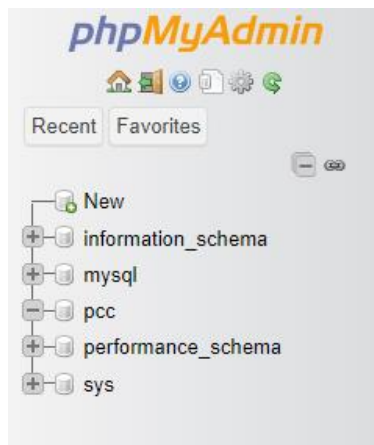
kita coba cek dengan mengakses lewat web browser dengan port 8090



login dengan username: **root**, password: **supersecret**



Dapat dilihat di sebelah kiri terdapat database bernama pcc



Latihan 2 (docker container)

1. Buat container dari image nginx
2. Beri nama "latihan nginx"
3. Publish container dengan port 8080 (cek dengan web browser)

Membuat docker image (Dockerfile)

1. Buat sebuah file dengan nama Dockerfile (huruf D harus kapital)
2. Instruksi FROM

Mengambil image yang sudah ada

```
FROM alpine
```

3. Instruksi RUN

Menjalankan perintah saat proses build

```
FROM alpine
```

```
RUN apk update && apk add --no-cache python3
```

Menjalankan perintah update repository apk alpine

4. Instruksi WORKDIR

```
FROM alpine
```

```
RUN apk update && apk add --no-cache python3
```

```
WORKDIR /app
```

Pada saat container dijalankan, maka akan langsung masuk ke direktori tujuan

5. Instruksi ADD/COPY

```
FROM alpine
```

```
RUN apk update && apk add --no-cache python3
```

```
WORKDIR /app
```

```
ADD script.py /app/
```

6. Instruksi CMD

Mengeksekusi command setelah fase build atau pada saat container dijalankan

```
FROM alpine
```

```
RUN apk update && apk add --no-cache python3
```

```
RUN mkdir /coba

WORKDIR /app

ADD script.py /app/

CMD ["python3", "script.py"]
```

7. Buat sebuah file dengan nama script.py untuk uji coba

```
for i in range(10):
    print("Halo Dunia!")
```

8. Jika digabungkan akan menjadi seperti berikut:

```
# Gunakan image alpine terbaru yang berukuran kecil
FROM alpine:latest

# Jalankan perintah update dan install paket yang
diperlukan
RUN apk update && apk add --no-cache python3

# Set folder kerja
WORKDIR /app

# Tambahkan file script.py ke dalam folder kerja
ADD script.py /app/

# Set perintah default saat container dijalankan
CMD ["python3", "script.py"]
```

9. Simpan file
10. Build dengan perintah **docker build -t coba .**
11. Kemudian jalankan **docker run -it coba**

```
pcc-class@network:~/coba$ docker run -it coba
Halo Dunia!
Halo Dunia!
Halo Dunia!
Halo Dunia!
Halo Dunia!
Halo Dunia!
Halo Dunia!
Halo Dunia!
Halo Dunia!
Halo Dunia!
```

Hasil outputnya sama seperti script python yang sudah kita buat.

Latihan 3

1. Buat Dockerfile di dalam direktori ini
2. Gunakan image nginx
3. Ubah tampilan web default nginx
4. Ubah tampilan bebas, pastikan bukan "Welcome to nginx"
5. Beri nama hasil image "latihan-dockerfile"
6. Jalankan container dengan nama "latihan03" dan publish ke port 8000
7. Test browsing dengan port 8000

#hint:

1. tempat menyimpan file html di nginx adalah /usr/share/nginx/html/
2. bisa dengan menghapusnya atau mengubah (bebas)
3. atau bisa timpa filenya dengan isi dari direktori mywebsite



PCC CLASS 2024