



**Project Based Internship**

## Docker as CI/CD Tools

**Pengenalan mengenai Docker sebagai  
CI/CD tools**

## Daftar Isi

A. Apa itu Docker	4
B. Docker vs Virtual Machine	5
C. Apakah Docker merupakan CI/CD tools	6
D. Konsep Docker	8
a. Instalasi Docker	8
b. Docker Image	9
c. Docker Container	10
d. Docker Hub	11
E. Implementasi Docker Pada Projek Vue	12
References	15



## A. Apa itu Docker

Docker adalah platform perangkat lunak yang memungkinkan Anda membuat, menguji, dan menerapkan aplikasi dengan cepat. Docker mengemas perangkat lunak ke dalam unit standar yang disebut kontainer yang memiliki semua yang diperlukan perangkat lunak agar dapat berfungsi termasuk pustaka, alat sistem, kode, dan waktu proses. Dengan menggunakan Docker, Anda dapat dengan cepat menerapkan dan menskalakan aplikasi ke lingkungan apa pun dan yakin bahwa kode Anda akan berjalan.

Contohnya adalah saat anda mendeploy program anda pada docker, maka ketika anda membagikan program anda pada orang lain, orang tersebut tidak harus dipusingkan untuk menginstal environment yang dibutuhkan untuk menjalankan program tersebut karena secara otomatis docker akan menyiapkan kebutuhan environment dari program tersebut. hal ini akan mempermudah developer untuk lebih fokus dalam melakukan penulisan kode saja dan tidak dipusingkan dengan deployment.



Jika diibaratkan dengan sebuah container yang sering kita lihat dipelabuhan, maka kita akan melihat container dapat dalam berbagai mode transportasi, baik kapal, kereta hingga trusk semua dapat mengangkut container, hal ini dikarenakan container sudah memiliki bentuk umum yang sudah dapat digunakan diberbagai mode transportasi tadi sehingga tidak perlu melakukan bongkar muat isi dari container ketika ingin mengangukutnya dengan mode transportasi yang berbeda.

Begitu pula dengan konsep container pada docker. Container pada docker digunakan untuk mengemas program kita dengan standarisasi yang sudah ditentukan. Oleh karena itu apabila container tersebut akan dijalankan pada sistem operasi lain / komputer lain maka tidak perlu melakukan penyesuaian pada environtement untuk dapat menjalankan program karena setup environtement sudah dikemas didalam container bersamaan dengan program anda.

## B. Docker vs Virtual Machine

Secara sekilas, jika dilihat mengenai cara kerja container pada docker sedikit mirip dengan konsep virtual machine. Namun apakah docker container merupakan sebuah virtual machine? Untuk menjawab hal tersebut kita akan bedah menganai apa itu virtual machine dan apa itu container.

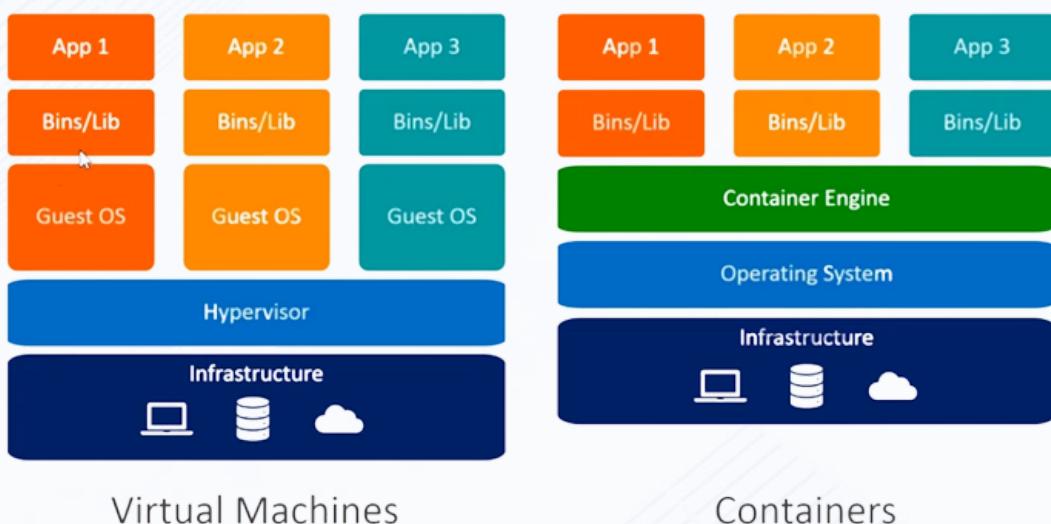
### a. Virtual Machine

Virtual machine adalah program perangkat lunak atau sistem operasi virtual yang bisa digunakan pada sebuah perangkat keras bersamaan dengan OS asli perangkat tersebut. VM akan diinstal pada hypervisor dan

memiliki fungsi layaknya duplikat dari komputer asli dengan sistem operasi berbeda.

#### b. Docker Container

Container merupakan suatu teknik yang dapat digunakan untuk menciptakan sistem yang terisolasi (isolated environment) pada level sistem operasi yang dijalankannya pada satu induk kernel (linux).



Jika anda perhatikan gambar diatas, ada beberapa perbedaan mendasar antara virtual machine dengan container. Untuk virtual machine sendiri dia akan berjalan pada hypervisor sehingga satu komputer bisa memiliki banyak sistem operasi. Sedangkan untuk container, dia akan berjalan pada sistem operasi dan bukan hypervisor sehingga dalam satu sistem operasi dapat memiliki banyak container.

## C. Apakah Docker merupakan CI/CD tools

Setelah anda memahami mengenai konsep docker, lantas apakah docker dapat dikategorikan sebagai CI/CD tools? Hal yang menarik adalah, ada

beberapa pendapat berbeda mengenai apakah docker dapat dikategorikan sebagai CI/CD tools atau tidak. Untuk itu mari kita bahas satu persatu.

a. Docker adalah CI/CD tools

Docker telah menjadi pengadopsi awal dalam Continuous Integration (CI) dan Continuous Deployment (CD). Dengan memanfaatkan integrasi yang tepat dengan mekanisme kontrol source code seperti GIT, Jenkins dapat memulai proses pembangunan setiap kali pengembang melakukan kodenya. Proses ini menghasilkan image Docker baru yang langsung tersedia di seluruh lingkungan. Dengan menggunakan image Docker, tim dapat membangun, berbagi, dan menerapkan aplikasi mereka dengan cepat. Berikut ini beberapa alasan lain mengapa docker dapat dikategorikan sebagai CI/CD tools :

**Support Every Platfotm:** Docker mendukung setiap platform untuk dijalankan. Itu dapat berjalan terus menerus dengan semua fitur continuous deployment dan continuous integration pada aplikasi.

**Coding and Testing:** Docker membantu pengembang perangkat lunak untuk menulis kode dan menguji dengan continous system. Itu sebabnya pengembang tidak perlu mengembangkan perangkat lunak dan memperbaruiinya lagi dan lagi.

**Otomation:** Docker dapat memperbarui perangkat lunak secara otomatis dan ini merupakan continous process.

b. Docker buka CI/CD tools

Docker bukan CI/CD tools. Docker adalah metodologi berkelanjutan untuk SDLC (Software Development Life Cycle). Jika Anda menjalankan CI/CD dengan container docker, Anda dapat menyebutnya CI/CD Docker. Dengan aplikasi docker container atau hub yang sempurna, Anda dapat

meningkatkan pengalaman keseluruhan alur kerja CI/CD tanpa mencapai batasnya, hal ini dikarenakan dengan digunakannya docker pada CI/CD pipeline, maka program dapat dijalankan diplatform manapun (fitur docker) dan secara otomatis dilakukan testing dan deployment (fitur CI/CD tools).

Jadi Docker bukan alat CI/CD itu sendiri tetapi dapat menjadi bagian dari proses CD di mana aplikasi Anda akan digunakan dalam container Docker dengan prosedur CD.

## D. Konsep Docker

### a. Instalasi Docker

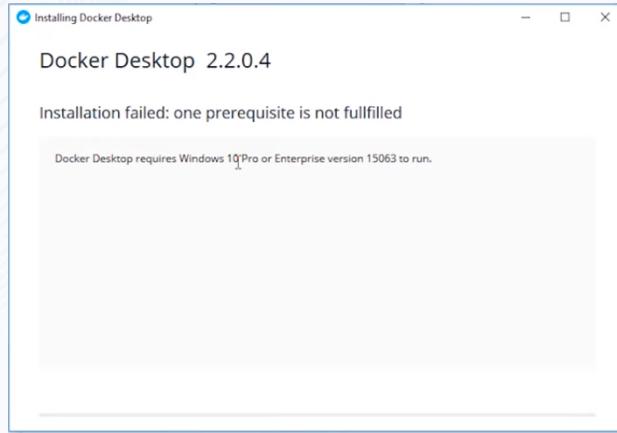
Anda dapat menginstall docker pada komputer anda melalui situs resmi dari docker (<https://docs.docker.com/>). Ada 3 sistem operasi yang dapat mendukung docker (Linux, Windows dan MacOS). Namun docker memiliki requirement yang cukup ketat untuk melakukan instalasi, apabila ada satu komponen pada requirement docker yang tidak terpenuhi pada komputer anda, maka instalasai akan gagal.

#### WSL 2 backend

- Windows 11 64-bit: Home or Pro version 21H2 or higher, or Enterprise or Education version 21H2 or higher.
- Windows 10 64-bit: Home or Pro 21H1 (build 19043) or higher, or Enterprise or Education 20H2 (build 19042) or higher.
- Enable the WSL 2 feature on Windows. For detailed instructions, refer to the [Microsoft documentation](#).
- The following hardware prerequisites are required to successfully run WSL 2 on Windows 10 or Windows 11:
  - 64-bit processor with [Second Level Address Translation \(SLAT\)](#)
  - 4GB system RAM
  - BIOS-level hardware virtualization support must be enabled in the BIOS settings. For more information, see [Virtualization](#).
- Download and install the [Linux kernel update package](#).

Namun apabila anda mendapati error seperti gambar diatas ketika anda melakukan instalasi docker pada komputer anda, maka anda dapat

mendownload docker melalui situs pihak ketiga (<https://runnable.com/docker/install-docker-on-windows-10>).



Jika anda telah selesai melakukan instalasi, maka anda dapat mengetikkan "docker --version" pada command prompt atau terminal anda untuk mengecek apakah docker benar-benar sudah terinstall.

```
C:\Users\ASUS>docker --version
Docker version 20.10.17, build 100c701
```

## b. Docker Image

Image Docker adalah file yang digunakan untuk mengeksekusi kode dalam container Docker. Image Docker bertindak sebagai seperangkat instruksi untuk membangun container Docker, seperti template. Image Docker juga bertindak sebagai titik awal saat menggunakan Docker. Image sebanding dengan snapshot di environment mesin virtual (VM).

Docker image dapat anda buat sendiri ataupun anda dapat menginstal image melalui docker hub. Berikut ini merupakan beberapa perintah yang dapat anda jalankan yang berhubungan dengan image :

- Menginstall image melalui docker hub

```
C:\Users\ASUS>docker pull NAMA_IMAGE:VERSI
```

- Menampilkan image yang dimiliki

```
C:\Users\ASUS>docker images
```

- Menghapus image

```
C:\Users\ASUS>docker image rm NAMA_IMAGE:VERSI
```

\*container pada image perlu diberhentikan dahulu sebelum dihapus

### c. Docker Container

Container merupakan pembagian proses yang dijalankan pada sebuah image, jadi satu buah image dapat memiliki banyak container. Berikut ini merupakan beberapa perintah yang dapat anda jalankan yang berhubungan dengan container :

- Membuat container

```
C:\Users\ASUS>docker create container --name NAMA_CONTAINER -p CUSTOM_PORT:DEFAULT_PORT NAMA_IMAGE:VERSI
```

contoh :

```
C:\Users\ASUS>docker create container --name vue_project -p 8080:8080 vueImage:latest
```

- Menampilkan container sedang berjalan

```
C:\Users\ASUS>docker container ps
```

- Menampilkan seluruh container (baik yang mati maupun dijalankan)

```
C:\Users\ASUS>docker container ls --all
```

- Menjalankan container

```
C:\Users\ASUS>docker container start NAMA_CONTAINER
```

- Menghapus container

```
C:\Users\ASUS>docker container rm NAMA_CONTAINER
```

\*container perlu diberhentikan dahulu sebelum dihapus

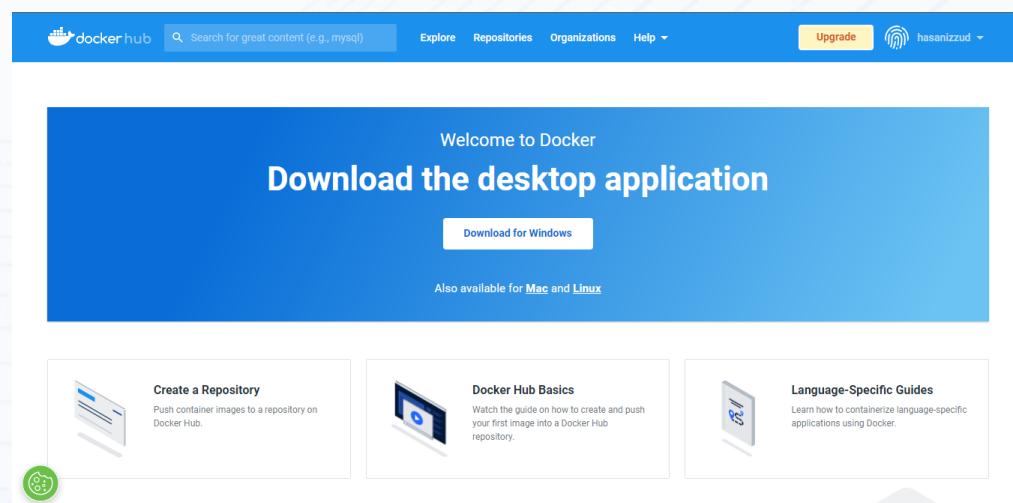
- Mengentikan container yang berjalan

```
C:\Users\ASUS>docker container stop NAMA_CONTAINER
```

## d. Docker Hub

Docker hub merupakan situs web komunitas resmi yang disediakan oleh docker. Selain anda dapat memperoleh image yang beredar di komunitas docker hub, anda juga dapat mempublish projek anda pada repository docker hub anda untuk nantinya digunakan oleh orang lain.

<https://hub.docker.com/>



## E. Implementasi Docker Pada Projek Vue

Untuk menjalankan projek vue anda dengan docker, anda perlu menyiapkan projek vue yang akan anda gunakan. Jika pada materi diatas mengenai image, anda dapat menginstall image yang anda dapat melalui docker hub, kali ini anda akan membuat image sendiri untuk wadah menjalankan container anda.

1. Siapkan Projek Vue anda.
2. Buat dockerfile pada folder projek anda.



3. Isi dockerfile dengan syntax berikut :

\*Syntax lengkap dapat anda lihat pada dokumentasi resmi vue js

<https://v2.vuejs.org/v2/cookbook/dockerize-vuejs-app.html>

```
# STEP 1 BUILD VUE PROJECT
FROM node:lts-alpine AS build-stage
WORKDIR /app
COPY package.json ./
RUN npm install
COPY .
RUN npm run build

# STEP 2 CREATE NGINX SERVER
FROM nginx:stable-alpine AS production-stage
COPY --from=build-stage /app/dist /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

4. Membuat image

docker build -t NAMA\_IMAGE .

```
PS D:\Project\Vue\tutorial_project> docker build -t dockervue .
```

\*pastikan nama image adalah lowercase dan berikan titik di akhir syntax

## 5. Tunggu hingga docker selesai menginstall image

```
[+] Building 204.5s (9/15)
=> sha256:213ec9aee27d8be045c6a92b7eac22c9a64b44558193775a1a7ff626352392b49 2.81MB / 2.81MB 191.0s
=> > extracting sha256:864b973d1bf1f32e9e7938e6f158ecf7fb503fbbe7799aacbf1fe853f506f398 5.1s
=> > extracting sha256:80fe61ad50f50607679112c84fb170dc5f9fc2130e875aa8d2d49d3a8fbdd 0.4s
=> > extracting sha256:e3887ab559e662a183c826648d3d51ba916a9be9119c1349b63ad6ca2c1c6c 0.0s
=> [production-stage 1/2] FROM docker.io/library/nginx:stable-alpine@sha256:98a1e37840fdf90f57df595dae8e27a198278170323744e13464b7f3a9 67.6s
=> > resolve docker.io/library/nginx:stable-alpine@sha256:98a1e37840fdf90f57df595dae8e27a198278170323744e13464b7f3a927562e 0.4s
=> sha256:98a1e37840fdf90f57df595dae8e27a198278170323744e13464b7f3a927562e 0.0s
=> sha256:5bc00c961f2ef45c7ed6057e555e095f6b700a9d2e7e824c5e3709f329a61d 1.57KB / 1.57KB 0.0s
=> > sha256:875b447b534ff2ff0860ac056a329b0f8f5afa93da1ae68e1948edd2e3fb13d 8.74KB / 8.74KB 0.0s
=> > sha256:213ec9aee27d8be045c6a92b7eac22c9a64b44558193775a1a7ff626352392b49 2.81MB / 2.81MB 19.9s
```

## 6. Cek apakah image sudah berhasil dibuat

```
PS D:\Project\Vue\tutorial_project> docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
dockervue      latest       4a41c6e019e3   32 seconds ago  24.4MB
```

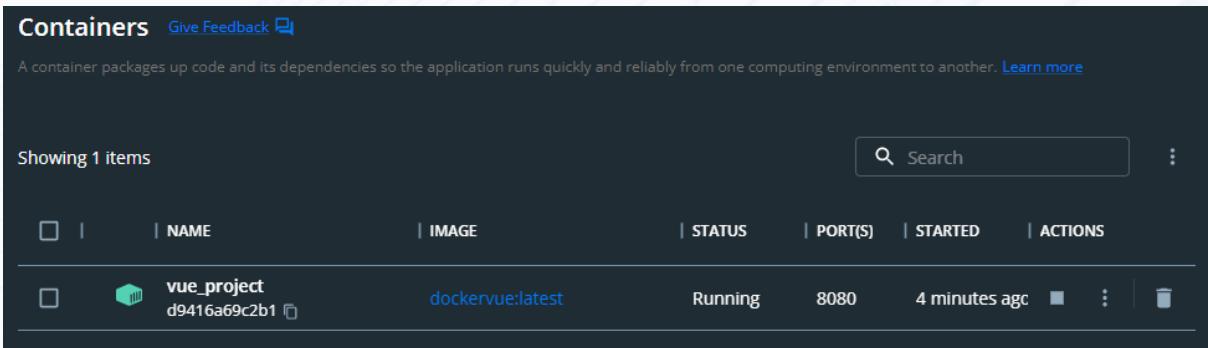
## 7. Buat container dengan langsung menjalankan.

```
PS D:\Project\Vue\tutorial_project> docker run -it --name vue_project -p 8080:80 dockervue
```

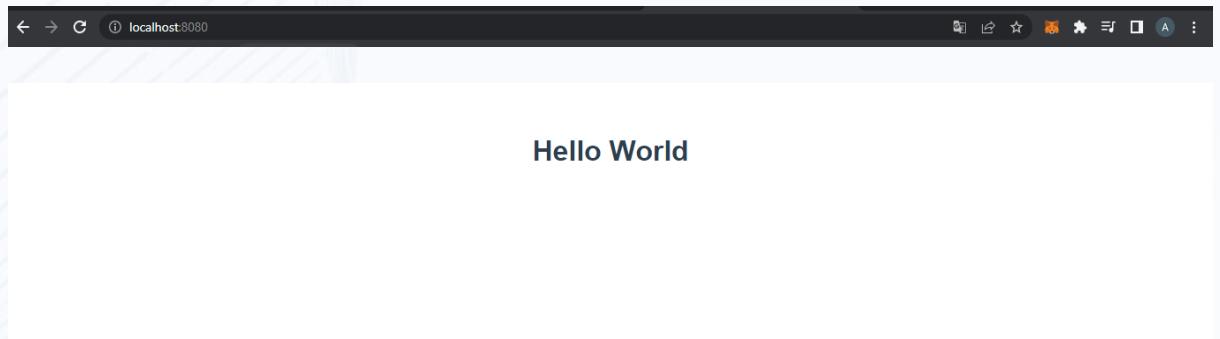
\*Ini merupakan shortcut untuk membuat dan menjalankan container, namun anda dapat melakukakannya secara manual dengan membuat container seperti biasa setelah itu menjalankannya

## 8. Cek apakah container sudah berjalan

```
PS D:\Project\Vue\tutorial_project> docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED        STATUS          PORTS          NAMES
d9416a69c2b1   dockervue     "/docker-entrypoint..."   3 minutes ago  Up 3 minutes   0.0.0.0:8080->80/tcp   vue_project
```



9. Anda dapat mengakses port seperti yang sudah anda tentukannya sebelumnya.



## References

[https://en.wikipedia.org/wiki/Docker\\_\(software\)](https://en.wikipedia.org/wiki/Docker_(software))

<https://opensource.com/resources/what-docker>

<https://www.ibm.com/cloud/learn/docker>

<https://www.cigniti.com/blog/need-use-dockers-ci-cd/>

<https://docs.docker.com/language/golang/configure-ci-cd/>

<https://www.linkedin.com/pulse/role-docker-cicd-pipeline-md-shoriful-isla>

m/

<https://v2.vuejs.org/v2/cookbook/dockerize-vuejs-app.html>