BINAR

# Docker

**Gold** - Chapter 6 - Topic 4

BINAR

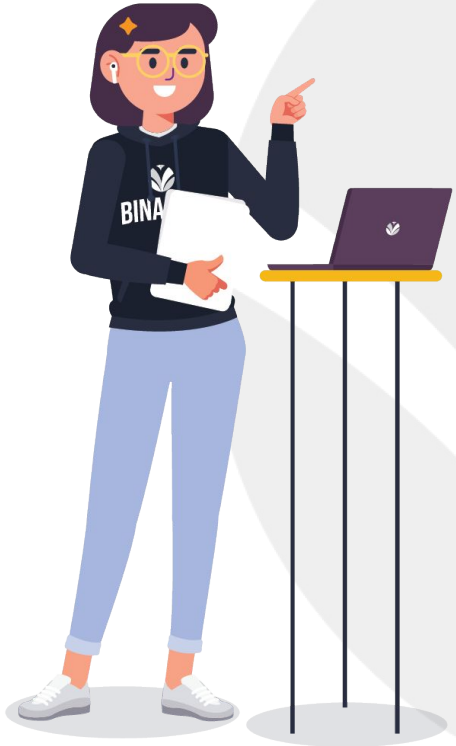
# Selamat datang di Chapter 6 Topic 4 online course Back End Java dari Binar Academy!

## Semangat baru untuk belajar hal baru 

Binarian, kita ketemu lagi di topik lanjutan setelah sebelumnya kita kupas tuntas tentang Java Logging.

Dari judul materi, kamu udah baca sekilas kalau hari ini kita belajar tentang **Docker.** Pokoknya mulai dari konsep containerization, setup Docker pada Heroku sampai melakukan deploy web apps sebagai Docker container hari ini kita akan kenalan dari dasar.

Yuk, langsung aja kita kepoin~



## Detailnya, kita bakal bahas hal-hal berikut ini:

- Konsep Containerization
- Cara membuat Docker Architecture
- Docker command untuk mengelola images dan containers
- Networks dan volumes pada Docker
- Cara melakukan setup Docker
- Deploy web apps sebagai Docker container

Docker merupakan salah satu platform untuk membuat container.

Karena docker dan container ini berkaitan erat, tentunya kita harus tahu dulu definisi container itu seperti apa.

Sebagai permulaan, kita mulai dari Containerization dulu, ya!

## Containerization itu apa yaa?

Containerization merupakan **pendekatan dalam menjalankan multiple instances (containers) pada satu Operating System (OS) yang sama**.

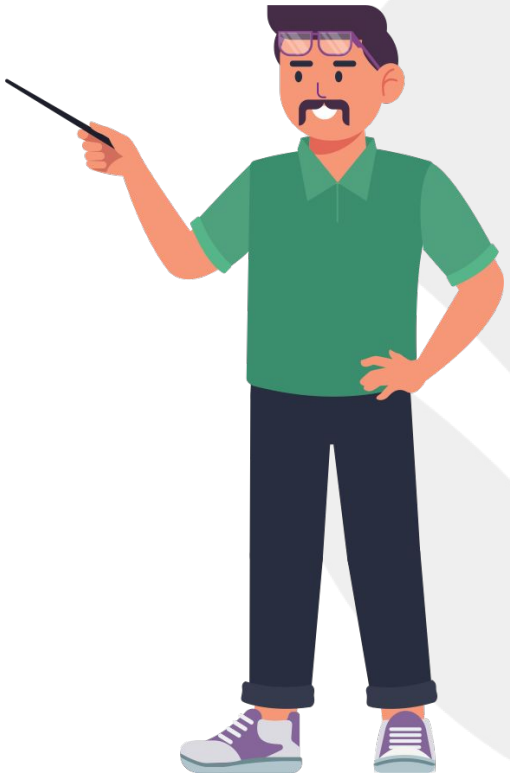
Di mana containers tersebut akan memiliki shared kernel.

Konsep containerization ini mirip kayak Virtual Machine (virtualization). Yaitu, ketika di satu Host OS kita bisa menjalankan berbagai aplikasi lain yang terisolasi dari Host kita.

Walaupun mirip kayak virtualization, perbedaan utama antara keduanya adalah pada virtualization kita bisa menghidupkan OS di dalam OS.

Kalau dilihat dari perbedaannya itu, kita bisa tahu juga bahwa secara resource dan performance, virtualization bakal lebih berat dibandingkan dengan **containerization.**

**Hal ini karena virtualization melakukan isolasi dari masing-masing aplikasi ke dalam virtual environment dengan shared kernel punya host**.

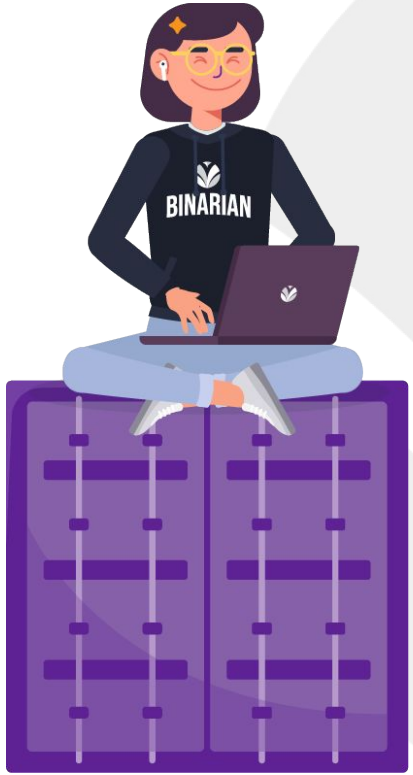


## Tapi, bukan berarti virtualization adalah teknologi yang buruk lho, ya!

Daripada menyebutnya buruk, kita bisa menggambarkan **virtualization sebagai teknologi yang unik tergantung dari tujuan pemakaiannya**.

Kedua teknologi di atas punya tujuan yang sedikit berbeda. Ibaratnya nih ya, kalau kita mau punya environment yang sangat terisolasi satu sama lain, maka sebaiknya kita menggunakan VM.

Kalau kita nggak perlu isolasi 100%, maka containerization bisa jadi pertimbangan.



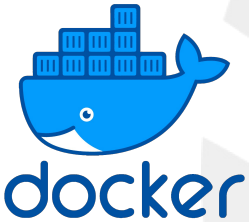
## Apa aja ya platform buat menjalankan container?

Dengan kebutuhan kita yang lebih cocok pakai container, di bawah ini ada beberapa platform yang bisa kamu pakai:

- **LXD (Linux Containers)**
- **Docker**
- **ACS (Azure Container Service)**
- **ECS (Amazon Elastic Container Service)**

Dari keempat platform container ini, yang bakalan kita pakai cuma Docker aja, ya!

Linux Containers





## Eh ada istilah baru nih, namanya Container Orchestration!

“Container Orchestration? Apaan tuh?”

Container Orchestration adalah sebuah tools yang bisa melayani host dalam sebuah group dan membentuk sebuah cluster.

Kenapa sih kita penting mempelajari ini? Yepp, Container Orchestration bisa membantu kita dalam memenuhi layanan kayak di bawah ini:

- Bisa Fault-Tolerant.
- Bisa discale up secara mudah dan on-demand process.
- Pakai resources secara maksimal (cont.)

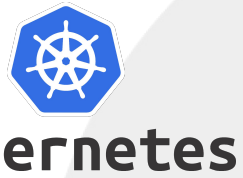
- Bisa menemukan & berkomunikasi secara otomatis pakai aplikasi yang kita bangun dalam satu cluster satu sama lain.
- Bisa diakses oleh seluruh dunia.
- Bisa update atau rollback tanpa adanya Downtime.

Dari manfaat yang udah dijabarin diatas, sekarang mulai kebayang kan arti penting Container Orchestration? Keren banget, kan?! 

## Apa aja ya orchestration platform itu?

Okey, secara lebih spesifik berikut adalah platform yang bisa kita pakai:

- Kubernetes
- EKS (Amazon Elastic Kubernetes Service)
- AKS (Azure Kubernetes Service)
- GKS (Google Kubernetes Engine)






Ternyata proses untuk membuat aplikasi nggak bisa satset satset langsung jadi, lho.

“Maksudnya?”

Ada konsep Docker Architecture yang perlu kita pelajarin dulu. Let’s go!

## Sekarang, kalau kita mau mengembangkan aplikasi butuh lebih dari sekadar menulis kode, gengs!

Kenapa? karena ada kompleksitas yang luar biasa dari proses pembuatan aplikasi itu sendiri.

Iya kompleks, soalnya perlu mikirin penggunaan berbagai bahasa, kerangka kerja, arsitektur, serta antarmuka terpisah antar service.

Dibayangin aja udah wakwaw ya, pikiran ini~

Tapi, apakah sampai situ aja perjuangan kita berakhir?

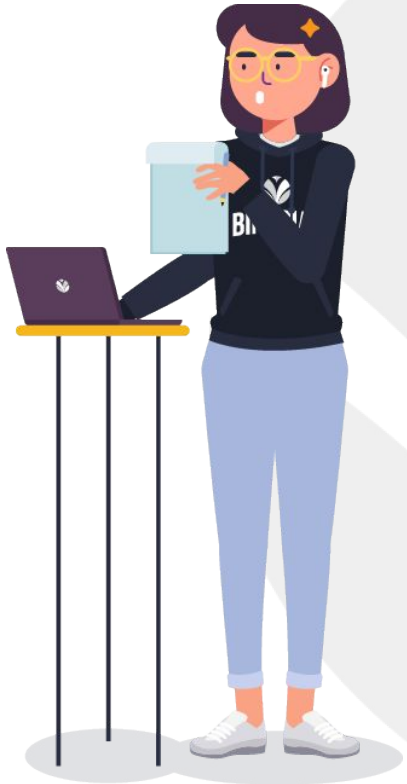
Tentu aja enggak, donk. Ada docker yang hadir untuk **menyederhanakan dan mempercepat alur kerja kita.**

Thanks to docker karena selain make it simple, doi sekaligus memberikan kebebasan kepada developer buat berinovasi dengan alat, techstack, dan deployment environment untuk setiap proyek.

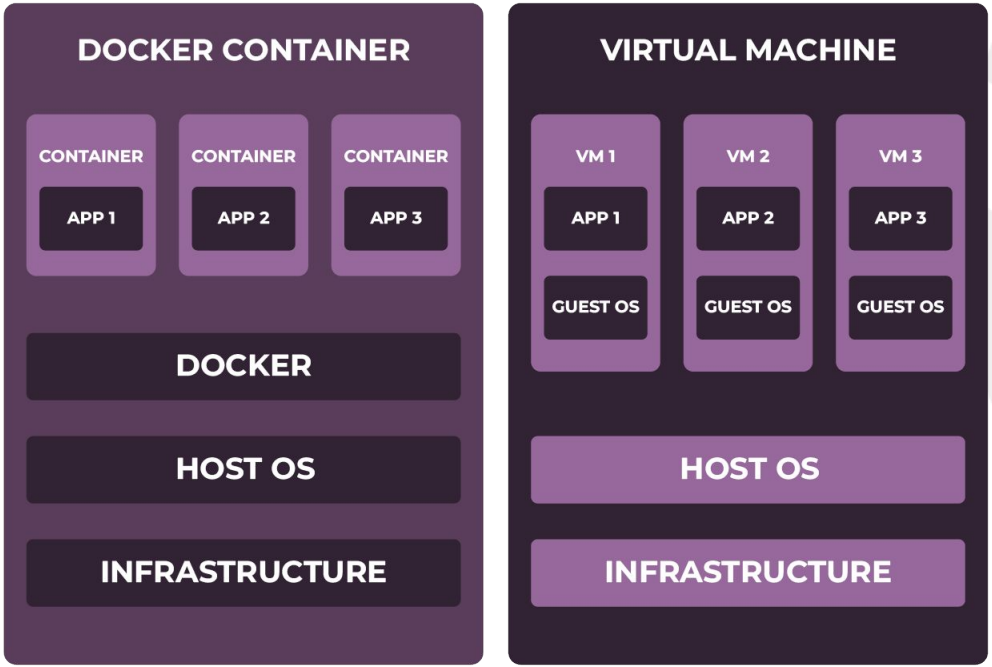
## Sampai sini, kita coba gabungin semua penjelasan tentang Docker dulu, nih~

Docker adalah sebuah **platform pembuat container yang punya komponen-komponen penyusun yang membuatnya bisa beroperasi**.

Container Docker bertindak mirip kayak mesin virtual atau virtual machine (VM). Bedanya, kalau VM bekerja dengan cara mem-virtualisasi hardware server, berarti container memvirtualisasi sistem operasi pada server.

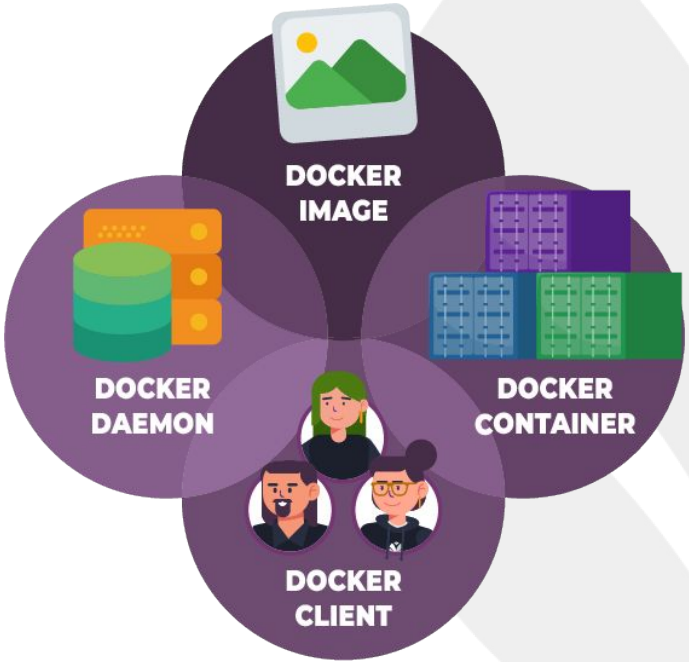


Di bawah ini ada perbandingan antara arsitektur Docker dan VM!



Nggak cuma itu aja, ketika ngomongin Docker, kamu nggak akan lepas dari istilah-istilah yang ada di bawah ini:

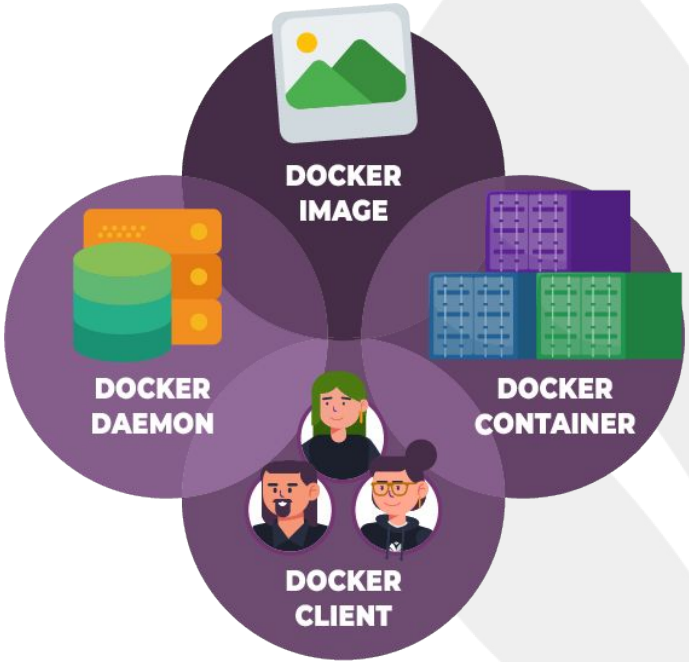
- **Docker image** adalah kumpulan file yang berisi informasi untuk membangun sebuah kontainer.
- **Docker container** adalah environment untuk mengemas aplikasi yang mencakup system tools, library, code, runtime dan konfigurasi.



- **Docker client** merupakan tempat user mengirimkan command kayak Docker run, build, dan pull pada Docker daemon.
- **Docker daemon** adalah tempat pengelolaan Docker image, container, network, dan volume.

  Daemon menyediakan command-line-interface (CLI) sisi client supaya user bisa berinteraksi dengan daemon lewat Docker API.

  Tugasnya yaitu menerima request dari Docker API yang bakal diproses selanjutnya oleh sistem.



Okeeey! Udah tahu ya kaitan container ke docker itu kayak gimana 

Selanjutnya kita bakal cari tahu tentang **Docker Commands to Manage Images and Containers.**

Nggak usah nunggu lama lagi, yuk kita meluncurrr~

## Penasaran sama command-nya? Coba cek link yang di bawah, ya!

Pada link berikut berisi list dari command-command buat mengelola images dan containers.

Silahkan diintip, gengs~

Docker Cheat Sheet

Nggak cuma HP aja nih yang punya network buat akses internet, Docker juga punya network!

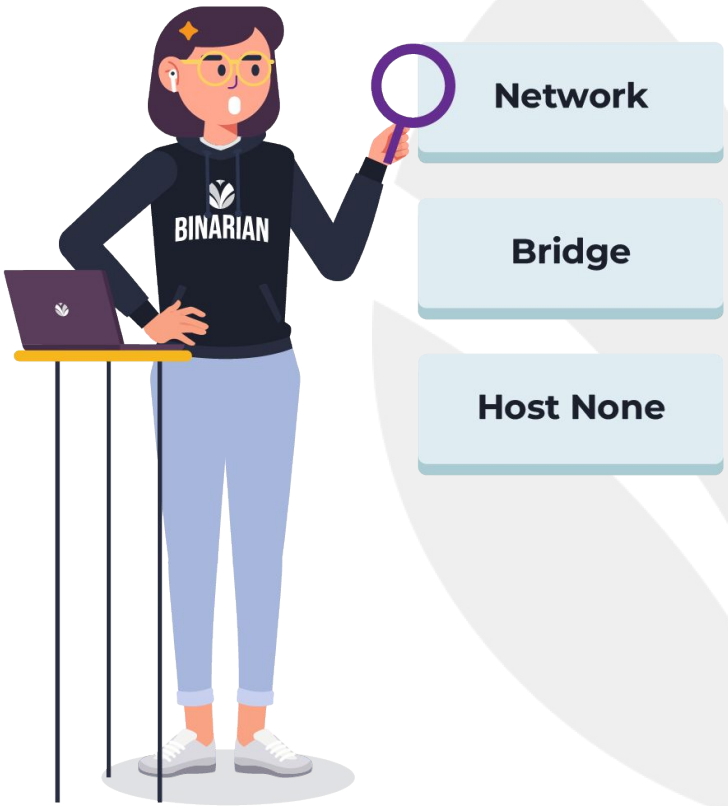
Oleh karena itu, mari kita masuk ke materi **Networks in Docker.**

Kita pantun dulu, berang-berang bawa tongkat, berangkat~

## Networks yang ada di Docker~

Sebagai service container, Docker punya 3 jenis **network**, **bridge**, dan **host none**. Dimana container disini baru diset menggunakan bridge network by default.

Oh ya, untuk mengaktifkan komunikasi antara beberapa wadah, kamu bisa bikin jaringan baru dan meluncurkan wadah di dalamnya.



Kenapa harus bikin jaringan baru?

Hal ini memungkinkan wadah untuk berkomunikasi satu sama lain ketika diisolasi dari wadah yang nggak terhubung ke jaringan.

Selain itu, jaringan baru juga memungkinkan upaya memetakan nama kontainer ke alamat IP mereka.

## Berikut adalah networks driver dari Docker~

- **Bridge**, menghubungkan container yang pakai nama bridge yang sama.
- **Host**, menghapus network isolation antara container dan Docker host secara langsung pakai network punya host.
- **Overlay**, menghubungkan beberapa Docker daemon bersama-sama dan memungkinkan swarm service berkomunikasi satu sama lain.

- **Macvlan**, memungkinkan menetapkan MAC address ke container dan membuatnya muncul sebagai perangkat fisik di network.
- **None**, menonaktifkan semua network. Biasanya digunakan bersama dengan custom network driver.
- **Network Plugins**, menginstall dan menggunakan third-party network plugin yang tersedia di Docker Hub atau dari third-party vendor.

Setelah mempelajari network dari docker kita bakal lanjut ke Volume in Docker yang isinya berupa cara menyimpan data.

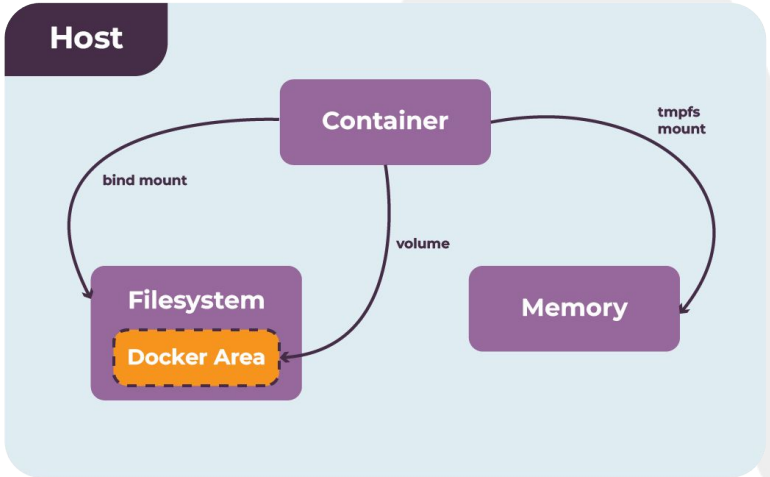
Inget ya, volume disini beda banget sama volume yang biasa kita pahami di konsep suara.

So, beli es naik angkot. Let's Cekidott!

## “Maksudnya volume in Docker itu gimana, sih?”

Volume Docker adalah mekanisme untuk **menyimpan data yang dihasilkan dan dipakai oleh container Docker**.

Kalau mount dan bind bergantung pada struktur direktori dan OS dari host, maka volume sepenuhnya dikelola sama Docker.



## Bukan sembarang cara, volume juga punya beberapa keunggulan nih, sob!

Dibandingkan mount dan bind, berikut adalah keunggulan dari volume:

- Lebih mudah buat di-backup atau dimigrasi daripada bind mount.
- Kamu bisa mengelola volume pakai perintah Docker CLI atau API Docker.
- Berfungsi di container Linux dan Windows (cont.)

- Bisa dibagikan dengan lebih aman di antara banyak container.
- Memungkinkan untuk menyimpan volume di remote host atau penyedia cloud dalam meng-enkripsi konten volume atau menambahkan fungsionalitas lainnya.

## Lanjut~

- Volume baru bisa punya konten yang telah diisi sebelumnya oleh sebuah wadah.
- Volume di Docker Desktop punya kinerja yang jauh lebih tinggi daripada bind mount dari host Mac dan Windows.

Gimana? Keren banget kan manfaatnya?!

Ini dia yang ditunggu-tunggu!

Kita bakal cari tahu gimana melakukan setup Docker di Dokku.

Udah penasaran banget, kan? Capcus kita mulai setupnya~

**Untuk melakukan containerization pada heroku dengan Spring Boot, kita bisa simak referensi berikut, ya!**

Silahkan menyimak referensinya, bestie~

Deploying a Spring Boot API to Railway using Maven and Docker

Ternyata nggak sesulit itu, kan?

Lanjut nih lanjut. Supaya pemahaman kamu makin ajib, abis ini kita coba melakukan **Deploy web apps sebagai docker container.**

Yuk!

## Ayo kita latihan! 

Kamu udah familiar sama basic dan fundamental containerization dan docker. Sekarang dengan bantuan facilitator atau teman sekelas, coba lakukan deployment web app.

Kamu bisa pakai aplikasi REST API pada challenge di chapter sebelumnya. Lalu deploy di laptop atau PC kamu dengan menginstallkan Docker.

Kalau kamu kepo sama langkah-langkahnya, kamu bisa simak di sini, yaaa~

Wow kamu udah sampe sini ternyata?! Kereeennn!

Kalau kita flashback lagi, Docker bisa digunakan untuk membuat, menjalankan, dan mengelola aplikasi menggunakan container.

Kira-kira pada case apa ya kita bisa menggunakan Docker pada aplikasi Java?

Nah, selesai sudah pembahasan kita di Chapter 6 topik 4 ini. Keren bwanged loch!

Selanjutnya, kita akan ngobrolin tentang **Deployment** yang makin serius tapi santai karena kita belajar semua dari dasar, hihi 

Sampai ketemu di topik selanjutnya~