1. Perbedaan Linux Kernel & Distro

Linux Kerner adalah inti dari sistem operasi Linux.

Distro Linux adalah sistem operasi lengkap yang dibangun di atas kernel Linux.Setiap distro dibangun demgan tujuan dan karakteristik yang berbeda beda.

1. Linux FHX

Linux FHX Adalah  standar untuk mendefinisikan struktur direktori dan konten sistem operasi Linux dalam bentuk hirarki.

1. Sistem Permission dan owner Linux

Sistem Permission Adalah Sistem pada Linux yang mengatur akses pengguna terhadap file dan direktori. Setiap file dan direktori memiliki tiga kategori utama: pemilik (**owner**), grup, dan lainnya (other), yang masing-masingnya memiliki kolom izin baca, tulis, dan eksekusi.

1. Prinsip Enkripsi pada SSH

Prinsip enkripsi pada SSH melibatkan 2 jenis enkripsi utama: simetris, asimetris.

Enkripsi Simetris digunakan untuk mengenkripsi data selama sesi SSH.

Enkripsi asimetris digunakan dalam proses pertukaran kunci (key exchange).

1. Perbedaan Antara Http dan Https

Perbedaan utama antara HTTP dan HTTPS terletak pada keamanannya.

HTTP (Hypertext Transfer Protocol) adalah protokol dasar untuk mentransfer data di web, namun tidak mengenkripsi data yang dikirim.

Sedangkan HTTPS (Hypertext Transfer Protocol Secure) adalah versi aman dari HTTP yang menggunakan enkripsi.

1. Docker OCI compliance standard

Docker dibangun berdasarkan standar Open Container Initiative (OCI), yang Dimana OCI Adalah  struktur tata kelola ringan dan terbuka yang dibentuk untuk menciptakan standar terbuka bagi format dan runtime container.

1. VM dan Container

VM:

Lingkungan virtual yang terisolasi, termasuk sistem operasi dan aplikasi. Setiap VM memiliki kernel sendiri, sehingga isolasinya kuat dan memungkinkan menjalankan beberapa OS yang berbeda di satu mesin fisik.

Kontainer:

Berbagi kernel sistem operasi host, sehingga lebih ringan dan efisien. Mereka hanya membungkus aplikasi dan dependensinya, tidak memerlukan sistem operasi lengkap.

1. Definisi dan Manfaat Image Layer pada docker

Definisi = Docker Image terdiri dari **lapisan-lapisan (layers)**. Setiap layer adalah hasil dari instruksi di **Dockerfile** (misalnya FROM, RUN, COPY). Layer bersifat **read-only** dan disusun bertumpuk → digabung menjadi satu image.  
  
Manfaat :

1. **Efisiensi penyimpanan** → Layer yang sama bisa dipakai ulang di banyak container, jadi hemat space.
2. **Caching build** → Jika ada perubahan di satu instruksi, Docker hanya membangun ulang layer setelahnya, sehingga build lebih cepat.
3. **Portabilitas** → Layer bisa didistribusikan dan dipakai di sistem lain (misalnya dari Docker Hub).
4. **Reusability** → Base image (misalnya ubuntu, node, python) bisa dipakai ulang untuk berbagai project.
5. Kegunaan dari penggunaan docker volume dan network beserta contohnya

Kegunaan Docker Volume :

1. Berbagi data antara Container.
2. Agar Data yang disimpan menjadi persinten sehingga container mati data tidak hilang

Contoh Docker Volume:

docker volume create mydata

docker run -d -v mydata:/var/lib/mysql mysql

Kegunaan Docker Network :

1. Memudahkan komunikasi antar container.
2. Isolasi jaringan antar aplikasi.

Contoh Docker Network :

docker network create mynetwork

docker run -d --name db --network mynetwork mysql

docker run -d --name app --network mynetwork node

1. Web server dan Reverse-proxy

Web server bertanggung jawab untuk melayani konten web kepada pengguna, sedangkan reverse proxy bertindak sebagai perantara antara klien dan server web, mengelola permintaan dan respons.

Web server Adalah Aplikasi yang melayani **permintaan (request) dari client/browser** dan mengirimkan **respon** (misalnya halaman HTML, gambar, API).

Kegunaannya:

1. Menyediakan konten web kepada user.
2. Menangani HTTP/HTTPS request dari browser.
3. Bisa juga melayani file statis (CSS, JS, gambar) atau meneruskan ke aplikasi backend.

Reverse-proxy Adalah Server yang berada **di depan satu atau lebih web server / aplikasi** yang menerima request dari client dan meneruskan ke server tujuan.

Kegunaannya

1. **Load Balancing** → membagi trafik ke beberapa server agar lebih stabil.
2. **Security** → menyembunyikan detail server backend dari client.
3. **SSL Termination** → mengelola sertifikat HTTPS di satu titik.
4. **Caching** → mempercepat respon dengan menyimpan hasil request.