TUGAS TERSTRUKTUR PERTEMUAN 4

STRUKTUR LAPORAN VIRTUALISASI & INSTALASI SERVER

Disusun oleh :

Putri Amalia

231011401598

TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PAMULANG

KATA PENGANTAR

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat rahmat dan karunia-Nya, saya dapat menyelesaikan laporan praktikum ini dengan baik. Laporan ini disusun sebagai salah satu tugas dalam mata kuliah yang berfokus pada teknologi virtualisasi, instalasi server, dan Docker. Dalam laporan ini, saya menjelaskan mengenai teori-teori dasar virtualisasi berbasis Virtual Machine (VM) dan Container, serta tahapan instalasi sistem operasi server dan Docker, berdasarkan pengalaman praktikum yang saya lakukan.

Laporan ini diharapkan dapat memberikan pemahaman yang lebih baik mengenai teknologi virtualisasi dan implementasinya dalam pengelolaan sistem server dan container. Saya menyadari bahwa laporan ini masih jauh dari sempurna, oleh karena itu, kritik dan saran yang membangun sangat saya harapkan untuk perbaikan di masa mendatang.

DAFTAR ISI

PENDAHULUAN

Seiring dengan perkembangan teknologi informasi yang semakin pesat, kebutuhan akan efisiensi dan fleksibilitas dalam pengelolaan infrastruktur TI semakin tinggi. Salah satu solusi yang banyak digunakan untuk mencapai hal ini adalah teknologi **virtualisasi**. Virtualisasi memungkinkan pemanfaatan perangkat keras secara lebih optimal dengan cara menjalankan beberapa sistem operasi atau aplikasi dalam lingkungan terisolasi pada satu perangkat keras fisik. Teknologi ini sangat penting dalam konteks pengelolaan server, cloud computing, dan data center.

Dua jenis virtualisasi yang umum digunakan adalah **Virtual Machine (VM)** dan **Container**. Virtual Machine mengandalkan **hypervisor** untuk menjalankan sistem operasi yang terisolasi pada sebuah mesin fisik, sementara container menggunakan kernel yang sama dengan sistem operasi host, menjadikannya lebih ringan dan efisien dalam penggunaan sumber daya.

Laporan ini bertujuan untuk memberikan pemahaman mengenai kedua jenis virtualisasi tersebut, serta tahapan-tahapan praktikum yang dilakukan dalam penginstalan **sistem operasi server** menggunakan **VirtualBox** dan pengelolaan **Docker**. Dengan laporan ini, diharapkan pembaca dapat lebih memahami penerapan teknologi virtualisasi dalam infrastruktur TI modern.

BAB I

TEORI VIRTUALISASI BERBASIS VIRTUAL MACHINE

Virtualisasi berbasis **Virtual Machine (VM)** adalah salah satu teknologi yang memungkinkan sebuah perangkat keras fisik untuk menjalankan beberapa sistem operasi secara bersamaan dalam lingkungan yang terisolasi, yang disebut sebagai mesin virtual. Teknologi ini sangat penting dalam pengelolaan infrastruktur TI, karena memungkinkan pemanfaatan sumber daya yang lebih efisien dan fleksibel. Setiap VM beroperasi secara independen dan memiliki sistem operasi, aplikasi, serta konfigurasi yang seolah-olah berjalan di perangkat keras fisik tersendiri. Hal ini menjadikan VM sangat cocok untuk berbagai keperluan, baik itu untuk pengujian perangkat lunak, pengelolaan server, maupun untuk menjalankan aplikasi yang membutuhkan konfigurasi berbeda dalam satu server.

Pada dasarannya, virtualisasi VM bekerja dengan menggunakan lapisan perangkat lunak yang disebut **hypervisor**. Hypervisor bertugas untuk mengelola dan mengalokasikan sumber daya perangkat keras (seperti CPU, memori, dan penyimpanan) ke setiap VM yang berjalan. Ada dua jenis hypervisor yang umum digunakan dalam virtualisasi berbasis VM:

1. **Type 1 (bare-metal)**: Hypervisor ini langsung beroperasi di atas perangkat keras fisik tanpa memerlukan sistem operasi host. Contoh dari hypervisor jenis ini adalah VMware ESXi, Microsoft Hyper-V, dan Xen. Hypervisor Type 1 cenderung lebih efisien dalam hal performa dan stabilitas karena langsung berinteraksi dengan perangkat keras.
2. **Type 2 (hosted)**: Hypervisor ini berjalan di atas sistem operasi host yang sudah ada, seperti VirtualBox dan VMware Workstation. Hypervisor Type 2 lebih mudah digunakan dan lebih cocok untuk keperluan pengembangan dan pembelajaran, tetapi performanya sedikit lebih rendah dibandingkan dengan Type 1.

Salah satu keunggulan utama dari virtualisasi berbasis VM adalah tingkat **isolasi** yang sangat baik antar mesin virtual. Setiap VM berjalan dalam lingkungan yang sepenuhnya terpisah, sehingga jika terjadi masalah atau kesalahan dalam satu VM, itu tidak akan mempengaruhi VM lainnya. Isolasi ini juga memberikan tingkat keamanan yang lebih tinggi karena setiap VM dapat dikonfigurasi dengan kebijakan dan pengaturan keamanan yang berbeda. Misalnya, sebuah VM dapat digunakan untuk menguji perangkat lunak yang tidak dapat dipercaya tanpa memengaruhi lingkungan produksi yang ada.

Namun, meskipun virtualisasi berbasis VM menawarkan keuntungan dalam hal isolasi dan fleksibilitas, ada beberapa kelemahan yang perlu diperhatikan. Salah satunya adalah **konsumsi sumber daya** yang relatif lebih besar. Karena setiap VM menjalankan sistem operasi penuh, maka setiap VM membutuhkan ruang penyimpanan dan memori yang lebih besar. Selain itu, proses booting VM cenderung memakan waktu lebih lama dibandingkan dengan metode virtualisasi lainnya, seperti container.

Dalam konteks infrastruktur data center atau cloud computing, virtualisasi berbasis VM memungkinkan perusahaan untuk mengoptimalkan penggunaan perangkat keras mereka. Beberapa VM dapat dijalankan dalam satu mesin fisik, yang memungkinkan pemanfaatan lebih maksimal dari sumber daya yang ada. Hal ini dapat menurunkan biaya operasional dan pemeliharaan perangkat keras. Selain itu, VM juga mempermudah dalam **pencadangan (backup)** dan **pemulihan (recovery)**, karena seluruh lingkungan dalam sebuah VM dapat dengan mudah dipindahkan, dikloning, atau dipulihkan.

Selain itu, virtualisasi berbasis VM juga memungkinkan penerapan **high availability** dan **disaster recovery**. Dalam skenario ini, mesin virtual dapat dipindahkan dari satu host fisik ke host fisik lainnya tanpa gangguan layanan, yang memastikan sistem tetap berjalan meskipun terjadi kerusakan pada perangkat keras fisik.

Secara keseluruhan, virtualisasi berbasis VM adalah solusi yang sangat powerful untuk berbagai macam keperluan, dari pengembangan perangkat lunak, pengujian aplikasi, hingga pengelolaan infrastruktur TI skala besar. Meskipun memiliki beberapa kelemahan, terutama dalam hal konsumsi sumber daya, manfaat yang ditawarkan dalam hal fleksibilitas, keamanan, dan pengelolaan yang lebih baik menjadikan virtualisasi VM sebagai pilihan utama dalam banyak aplikasi teknologi modern.

BAB II

TEORI VIRTUALISASI BERBASIS CONTAINER

Virtualisasi berbasis **container** adalah teknologi yang memungkinkan aplikasi berjalan dalam lingkungan terisolasi di atas sistem operasi yang sama. Berbeda dengan virtualisasi berbasis Virtual Machine (VM), di mana setiap mesin virtual menjalankan sistem operasi lengkap, container hanya mengemas aplikasi bersama dependensinya (seperti pustaka atau library), sementara kernel sistem operasi host tetap digunakan bersama-sama oleh semua container. Pendekatan ini menjadikan container jauh lebih ringan dan efisien dalam penggunaan sumber daya dibandingkan dengan VM.

Container pertama kali populer dengan adanya proyek **Docker**, yang memungkinkan pembuatan, pengelolaan, dan distribusi aplikasi dalam container secara mudah. Docker memungkinkan pengembang untuk menjalankan aplikasi dalam wadah yang konsisten di berbagai lingkungan, mulai dari komputer pengembang hingga server produksi. Dengan demikian, aplikasi yang dibangun dalam container dapat berjalan dengan cara yang sama di berbagai platform tanpa perlu memikirkan perbedaan lingkungan, seperti sistem operasi atau versi pustaka yang terinstal.

Keunggulan Virtualisasi Berbasis Container

Salah satu keunggulan utama dari virtualisasi berbasis container adalah **efisiensi sumber daya** yang tinggi. Berbeda dengan VM yang memerlukan sistem operasi lengkap, container hanya membutuhkan aplikasi dan pustaka yang dibutuhkan untuk menjalankan aplikasi tersebut, sementara sistem operasi host sudah digunakan bersama. Hal ini mengurangi overhead dan memungkinkan lebih banyak container dijalankan dalam satu mesin fisik dibandingkan dengan VM.

Selain itu, container sangat **cepat** dalam hal proses mulai (startup). Karena container tidak perlu memuat sistem operasi lengkap, proses untuk menjalankan aplikasi dalam container hanya memerlukan waktu yang sangat singkat dibandingkan dengan VM yang perlu memulai sistem operasi terlebih dahulu.

Container juga menawarkan **portabilitas** yang sangat tinggi. Aplikasi yang berjalan dalam container dapat dipindahkan dan dijalankan di mana saja, baik di laptop pengembang, server internal, hingga cloud public atau private. Dengan menggunakan Docker, misalnya, aplikasi yang telah dibangun dalam container dapat dijalankan di berbagai sistem operasi yang mendukung Docker, seperti Linux, macOS, dan Windows. Hal ini membuat proses pengembangan dan deployment aplikasi menjadi lebih efisien dan tanpa hambatan terkait lingkungan.

Isolasi dan Keamanan dalam Container

Meskipun container menawarkan keuntungan dalam hal efisiensi, ia memiliki **tingkat isolasi yang lebih rendah** dibandingkan dengan VM. Karena container berbagi kernel yang sama dengan sistem operasi host, mereka tidak sepenuhnya terisolasi satu sama lain seperti halnya VM yang berjalan pada kernel yang berbeda. Ini berarti jika terjadi celah keamanan pada salah satu container, celah tersebut dapat berpotensi mempengaruhi container lain atau bahkan sistem operasi host itu sendiri.

Namun, Docker dan teknologi container lainnya telah mengimplementasikan berbagai mekanisme untuk meningkatkan isolasi dan keamanan, seperti penggunaan **namespace** dan **cgroups** untuk memisahkan ruang dan membatasi sumber daya antar container. Namespace memungkinkan container untuk memiliki pandangan terisolasi tentang sistem, sementara cgroups memungkinkan pembatasan dan pengelolaan penggunaan sumber daya, seperti CPU dan memori, pada setiap container.

Container dalam Konteks Microservices

Salah satu penerapan utama dari teknologi container adalah dalam arsitektur **microservices**. Microservices adalah pendekatan pengembangan perangkat lunak di mana aplikasi dibagi menjadi serangkaian layanan-layanan kecil yang berjalan secara independen dan berkomunikasi satu sama lain melalui API. Container sangat cocok untuk arsitektur microservices karena memungkinkan setiap layanan dijalankan dalam container terpisah, dengan dependensinya masing-masing, yang membuat pengelolaan aplikasi menjadi lebih mudah dan skalabel.

Setiap container dalam arsitektur microservices dapat di-deploy, diperbarui, dan diskalakan secara independen tanpa mengganggu layanan lainnya. Hal ini memungkinkan pengembangan yang lebih cepat dan fleksibel, serta mempermudah penerapan **continuous integration** dan **continuous delivery (CI/CD)**.

Pengelolaan Container dengan Docker

Docker adalah platform yang paling populer untuk mengelola container. Dengan Docker, pengembang dapat dengan mudah membuat, mengonfigurasi, dan menjalankan container menggunakan file **Dockerfile** yang berisi instruksi untuk membangun container. Docker juga menyediakan **Docker Hub**, sebuah repositori online untuk menyimpan dan berbagi gambar (image) container yang telah dibuat, yang memungkinkan tim pengembang untuk berbagi aplikasi dalam bentuk container.

Docker Compose adalah alat yang berguna untuk mengelola aplikasi multi-container. Dengan Docker Compose, pengguna dapat mendefinisikan dan menjalankan aplikasi yang terdiri dari beberapa container yang saling berinteraksi, misalnya aplikasi web yang terhubung dengan database. Konfigurasi untuk seluruh aplikasi dapat disimpan dalam satu file YAML, yang memungkinkan replikasi dan penyebaran aplikasi dengan mudah.

Kelemahan Virtualisasi Berbasis Container

Meskipun teknologi container menawarkan banyak keuntungan, ada beberapa kelemahan yang perlu diperhatikan. Salah satu kelemahannya adalah **isolasi yang lebih rendah** dibandingkan

dengan VM. Seperti yang telah disebutkan sebelumnya, karena container berbagi kernel dengan sistem operasi host, mereka tidak sepenuhnya terisolasi. Ini mungkin menjadi masalah dalam konteks aplikasi yang membutuhkan tingkat keamanan yang sangat tinggi atau aplikasi yang harus berjalan di lingkungan yang sangat terpisah.

Selain itu, meskipun container lebih ringan daripada VM, mereka masih memiliki **pengelolaan sumber daya** yang lebih terbatas. Ketika aplikasi dalam container memerlukan sumber daya yang sangat besar atau kompleks, pengelolaan sumber daya menjadi lebih sulit, terutama pada aplikasi yang memerlukan akses langsung ke perangkat keras atau siste

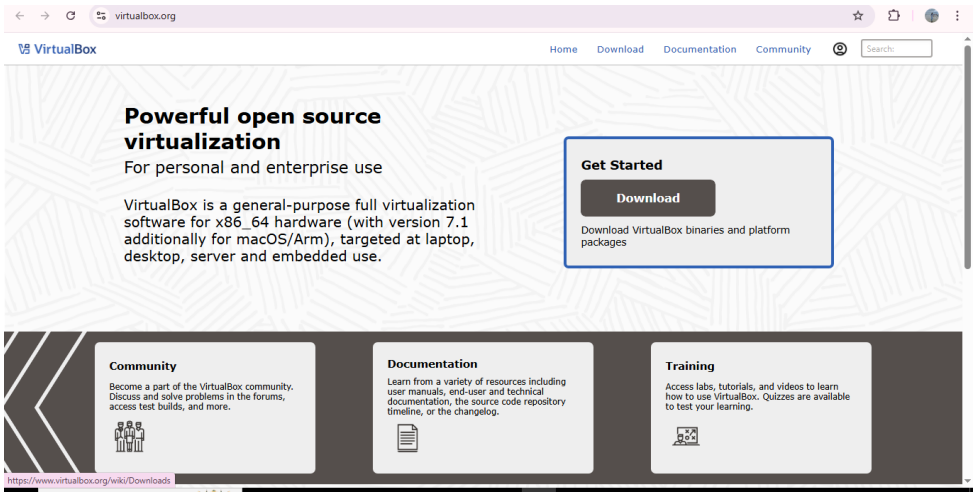
BAB III

TAHAPAN INSTALASI SISTEM OPERASI SERVER MENGGUNAKAN VIRTUALBOX

1. Download & Install VirtualBox

   Link: https://www.virtualbox.org

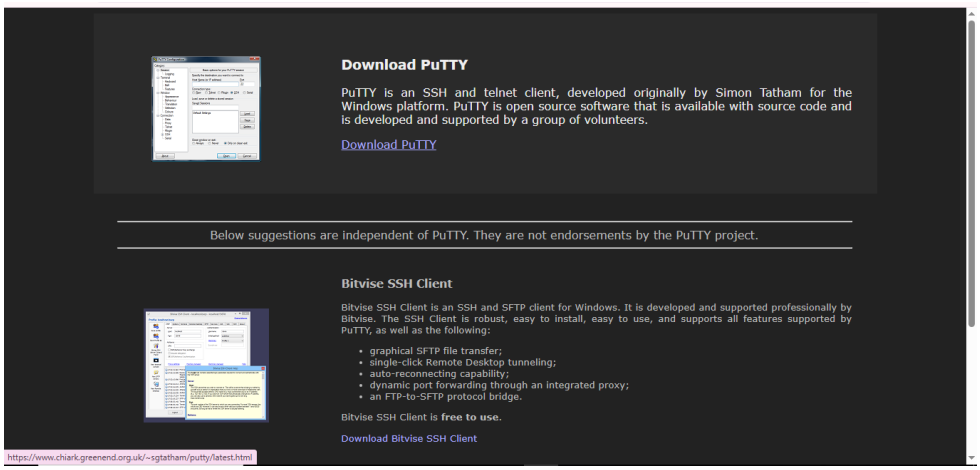
   - Kunjungi situs resmi dan unduh instalasi VirtualBox sesuai dengan sistem operasi yang digunakan.
   - Ikuti instruksi instalasi hingga selesai.



2. Install PuTTY di Windows

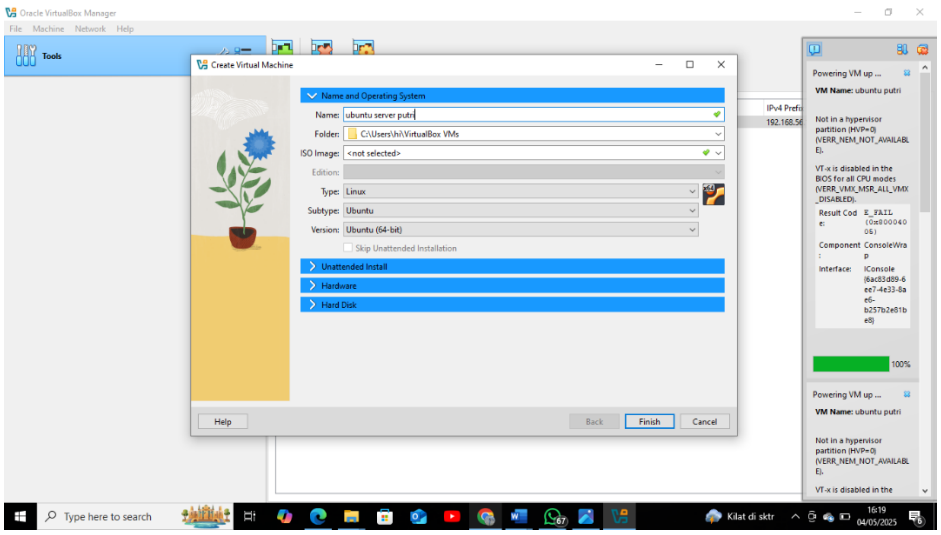
   Link: https://www.putty.org

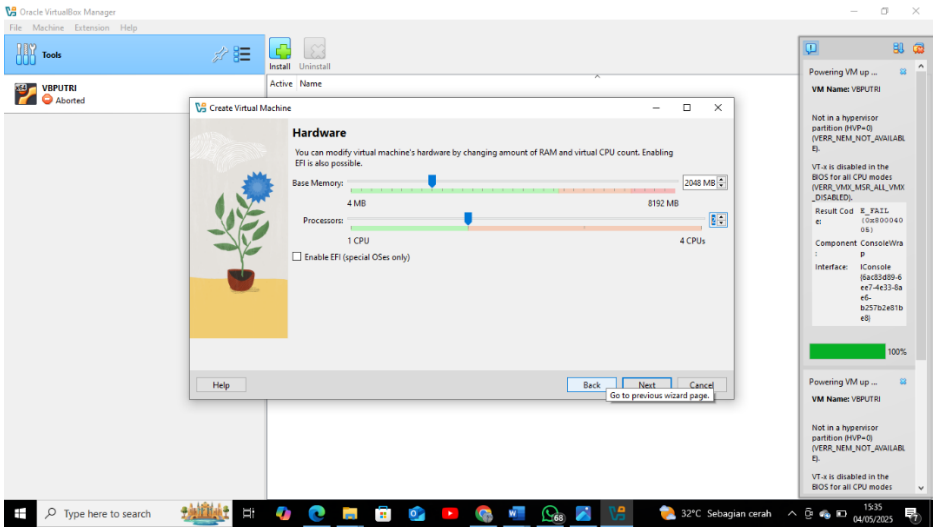
   - Unduh dan instal Putty untuk mengakses SSH.
   - Pastikan Putty terpasang dengan benar untuk melakukan koneksi remote ke server Ubuntu.

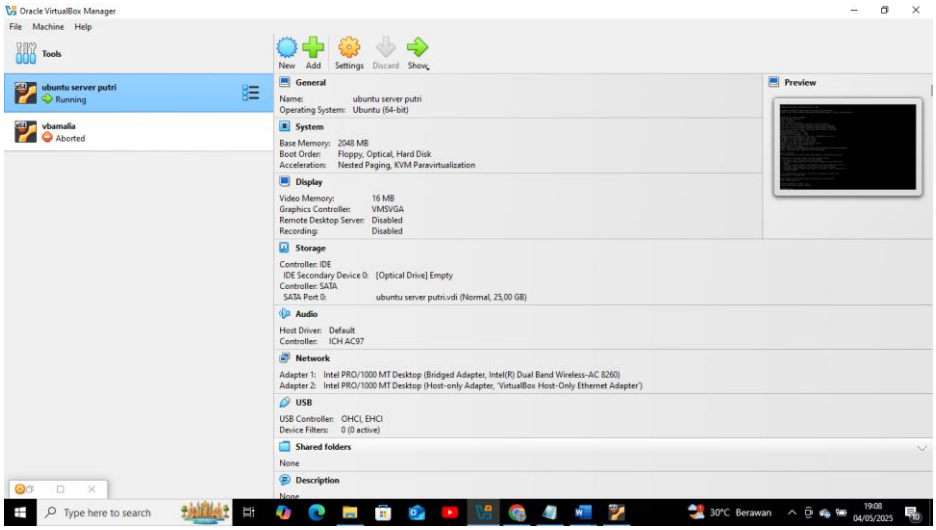


3. Install Sistem Operasi Server (Ubuntu Server, BUKAN Desktop)

- Tentukan alokasi sumber daya, seperti CPU, RAM, dan penyimpanan untuk VM.
- Install sistem operasi Ubuntu Server sesuai petunjuk di layar.
- Jika sudah diinstal maka buka aplikasi VirtualBox
- Setelah itu buat mesin virtual baru dan pilih Ubuntu Server sebagai sistem operasi. , dengan cara klik **NEW**
- Kemudian isi seperti di bawah ini





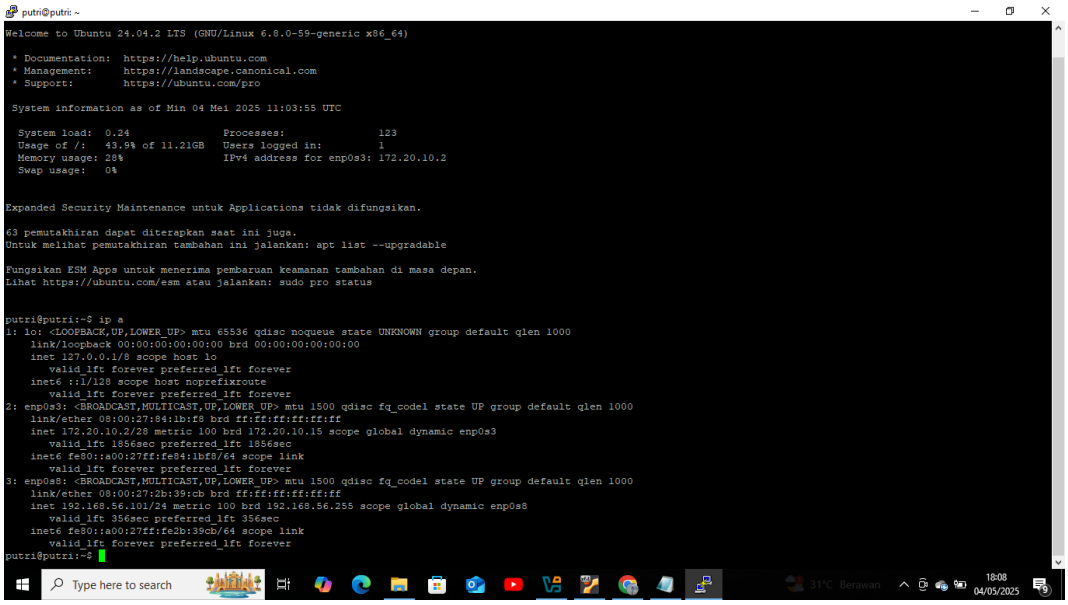


- Jika semuanya sudah sesuai seperti gambar ke 3, maka selanjutnya adalah klik **START** untuk memulai install Ubuntu di aplikasi Virtual Mechine

- Jika sudah nanti akan ada pilihan, pilih "**Try or Install Ubuntu Server"**
- Setelah itu pilih bahasa, jika sudah nanti ikuti langkah selanjutnya

4. Remote server via SSH

   - Cek Ip di Ubuntu Server



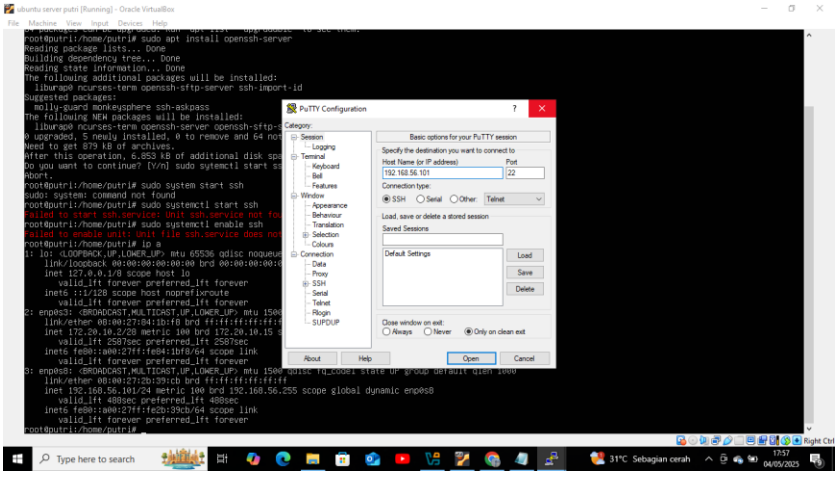
   - Install SSH server

     Ketikkan:
     sudo apt update
     sudo apt install openssh-server

   - Cek status

     Ketikkan:
     sudo systemctl status ssh

   - Remote via:
     PuTTY (Windows): masukkan IP, port 22

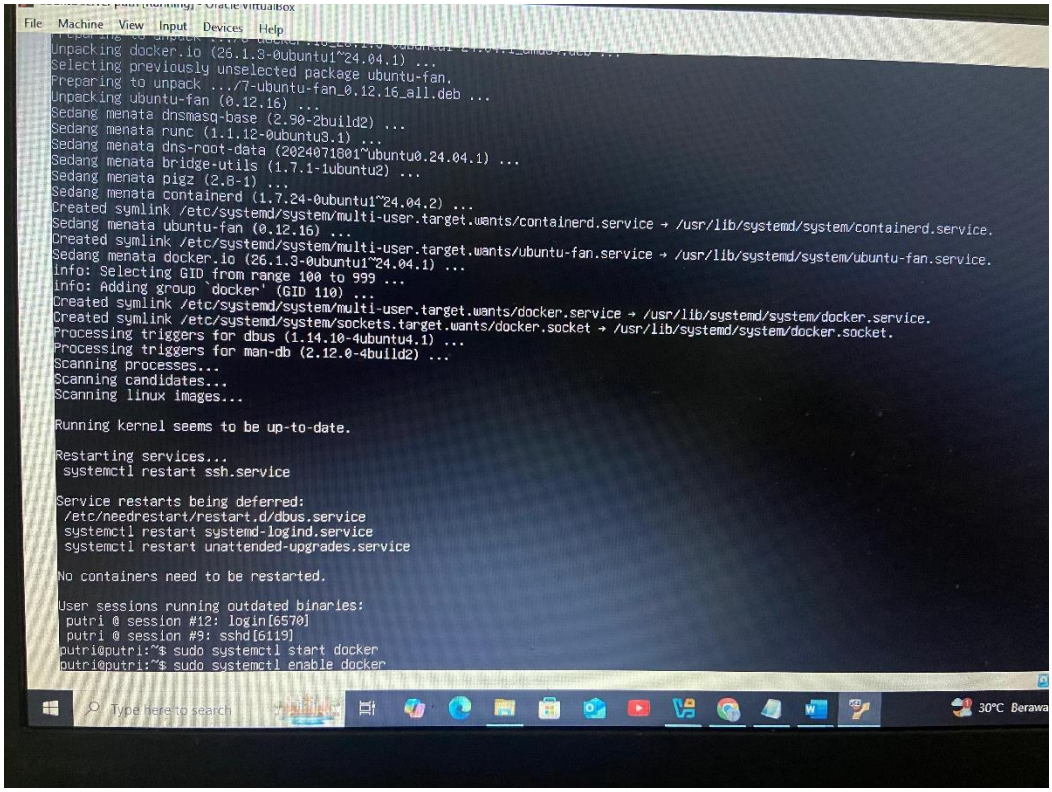


BAB IV

TAHAPAN INSTALASI DOCKER & TESTING

1. Instalasi Docker di Ubuntu Server

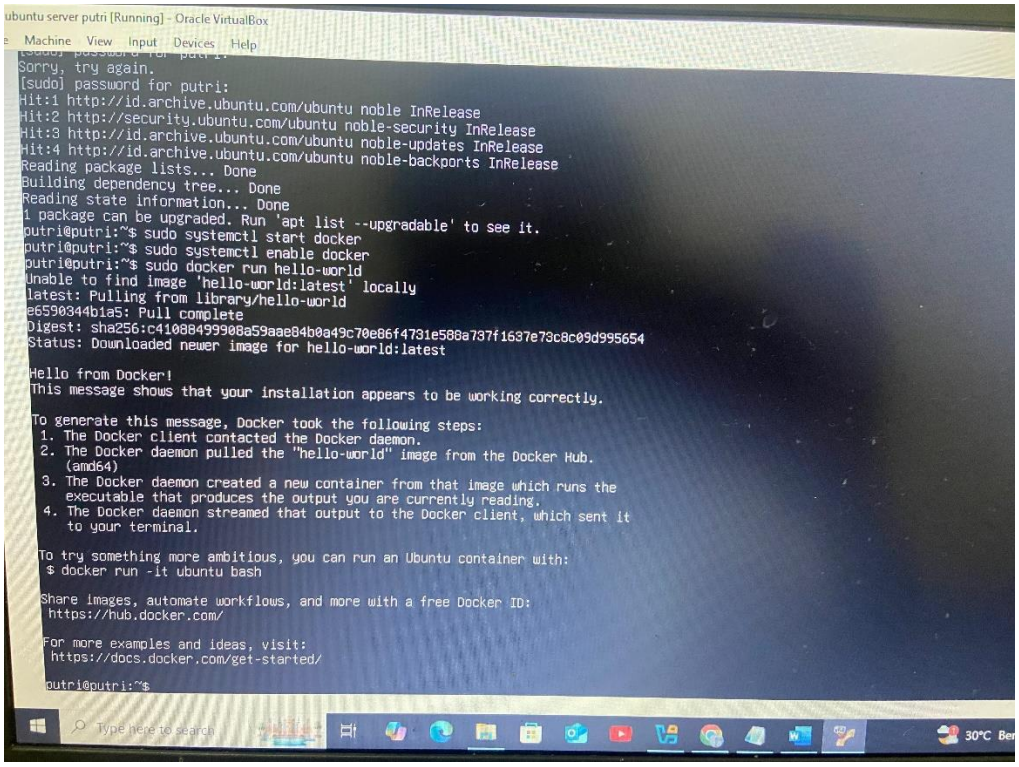
    - Gunakan perintah sudo apt-get update dan sudo apt-get install docker.io untuk menginstal Docker di Ubuntu Server.
    - Pastikan Docker berjalan dengan menggunakan perintah sudo systemctl start docker dan sudo systemctl enable docker.



2. Testing Docker

    - Jalankan perintah `docker --version` untuk memverifikasi apakah Docker berhasil terinstal.

- Coba jalankan perintah `docker run hello-world` untuk memastikan Docker dapat menarik dan menjalankan container dengan benar.



DAFTAR PUSTAKA

Docker, Inc. (2020). "What is Docker?" *https://www.docker.com/what-docker*.

Doyle, M. (2019). *Virtualization Essentials*. Wiley Publishing.

Harel, M. (2018). "Introduction to Virtualization Technologies." *Journal of Computing & Information Technology*, 26(2), 45-56.

Larsen, S. & Reinders, L. (2017). *The Docker Book: Containerization is the New Virtualization*. CreateSpace Independent Publishing Platform.

Merkel, D. (2014). "Docker: Lightweight Linux Containers for Consistent Development and Deployment." *Linux Journal*, 2014(239), 1-6.

Mouat, A. (2015). *Docker: Up & Running: Shipping Reliable Containers in Production*. O'Reilly Media.

Oracle Corporation. (2020). "Oracle VirtualBox User Manual." *https://www.virtualbox.org/manual*.

VMware, Inc. (2020). "VMware vSphere: Virtualization for the Modern Data Center." *https://www.vmware.com/products/vsphere.html*.