

LAPORAN PROGRES 2

Containerization dengan Docker

(Instalasi Docker)

Disusun untuk memenuhi Tugas Akhir Mata Kuliah Sistem Operasi

Dosen Pengampu:

Ferdi Chahyadi, S.Kom., M.Cs



Disusun oleh:

Akbar Risky Lingga	2401020003
Dzaky Ribal Faiz	2401020035
Muhammad Al-Fikry Akbar	2401020031
Al Adhlu sodri niwrad	2401020015

PRODI TEKNIK INFORMATIKA

FAKULTAS TEKNIK DAN TEKNOLOGI KEMARITIMAN

UNIVERSITAS MARITIM RAJA ALI HAJI

2025/2026

BAB I

PENDAHULUAN

1.1 Latar Belakang Progres

Laporan ini merupakan dokumentasi awal dari proyek mata kuliah Sistem Operasi yang berfokus pada penerapan teknologi container menggunakan Docker. Docker merupakan platform containerization yang memungkinkan aplikasi dijalankan secara terisolasi beserta seluruh dependensinya, sehingga mampu menciptakan lingkungan yang konsisten dan efisien pada berbagai sistem.

Pada progres pertama ini, kegiatan difokuskan pada proses instalasi Docker sebagai tahap fundamental sebelum masuk ke tahapan lanjutan, seperti pembuatan Dockerfile, Docker image, serta pengelolaan container. Instalasi Docker menjadi langkah penting karena keberhasilan seluruh proses containerization sangat bergantung pada kesiapan lingkungan sistem dan konfigurasi Docker yang digunakan.

Selain itu, proses instalasi Docker juga bertujuan untuk memberikan pemahaman awal kepada mahasiswa mengenai keterkaitan antara sistem operasi, kernel, dan teknologi container. Melalui instalasi ini, mahasiswa diperkenalkan pada konsep dasar virtualisasi ringan yang memanfaatkan fitur sistem operasi, seperti namespaces dan control groups (cgroups), untuk mengelola resource dan isolasi proses.

Dengan terlaksananya progres instalasi Docker ini, diharapkan sistem siap digunakan untuk tahap pengembangan selanjutnya, sekaligus memberikan dasar pemahaman yang kuat mengenai peran sistem operasi dalam mendukung teknologi container..

1.2 Tujuan Progres 2

- Mengetahui dan memahami konsep dasar teknologi container menggunakan Docker.
- Melakukan instalasi Docker pada sistem operasi sebagai tahap awal penerapan containerization.
- Memastikan Docker berhasil terpasang dan dapat dijalankan dengan baik pada sistem.
- Memahami keterkaitan antara sistem operasi dengan Docker, khususnya peran kernel dalam menjalankan container.
- Menyiapkan lingkungan sistem yang siap digunakan untuk tahap selanjutnya, seperti pembuatan Dockerfile dan Docker image.

1.3 Ruang Lingkup Progres 2

Progres 2 difokuskan pada tahap lanjutan penggunaan Docker, yaitu menjalankan container dari Docker image yang telah dibuat sebelumnya serta melakukan pengaturan penggunaan resource sistem. Pada tahap ini, Docker dimanfaatkan untuk mengelola alokasi CPU dan memori agar container dapat berjalan secara terkontrol dan tidak melebihi batas sumber daya yang tersedia pada sistem. Kegiatan ini bertujuan untuk memberikan pemahaman praktis mengenai manajemen resource pada lingkungan container berbasis sistem operasi.

Adapun ruang lingkup kegiatan pada Progres 2 meliputi:

- Menjalankan container Docker menggunakan Docker image yang telah dibuat pada progres sebelumnya.
- Melakukan pengaturan dan pembatasan penggunaan resource CPU pada container.
- Melakukan pengaturan dan pembatasan penggunaan resource memori (RAM) pada container.
- Menerapkan mekanisme control groups (cgroups) yang disediakan oleh Docker untuk manajemen resource.
- Melakukan pengujian terhadap container dengan resource yang telah dibatasi.

- Melakukan pengamatan dan analisis awal terhadap performa aplikasi di dalam container.

BAB II

Pembahasan

2.1 Proses Instalasi Docker

Pada tahap awal progres ini, dilakukan instalasi Docker pada sistem operasi Fedora Linux. Instalasi diawali dengan menambahkan repository resmi Docker ke dalam sistem menggunakan perintah `dnf config-manager`. Penambahan repository ini bertujuan agar sistem dapat mengunduh paket Docker versi stabil langsung dari sumber resmi, sehingga instalasi lebih aman dan terjamin kompatibilitasnya.

Setelah repository berhasil ditambahkan, proses instalasi Docker dilakukan menggunakan perintah `dnf install` yang mencakup beberapa komponen utama, seperti `docker-ce`, `docker-ce-cli`, `containerd.io`, serta plugin pendukung seperti `docker-compose` dan `docker-buildx`. Berdasarkan hasil instalasi, seluruh paket berhasil terpasang tanpa error, yang ditandai dengan status Complete pada proses instalasi.

```
13:16:45 akbarrzk@fedora ~
$ sudo dnf config-manager addrepo --from-repofile=https://download.docker.com/linux/fedora/docker-ce.repo
https://download.docker.com/linux/fedora/docker-ce.rep 100% | 10.0 KiB/s | 811.0 B | 00m00s

13:17:29 akbarrzk@fedora ~
$ sudo dnf install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Updating and loading repositories:
  Docker CE Stable - x86_64                                100% | 46.1 KiB/s | 18.3 KiB | 00m00s
Repositories loaded.
Package Arch Version Repository Size
Installing:
containerd.io x86_64 2.2.1-1.fc43 docker-ce-stable 120.0 MiB
docker-buildx-plugin x86_64 0.30.1-1.fc43 docker-ce-stable 77.9 MiB
docker-ce x86_64 3:29.1.3-1.fc43 docker-ce-stable 93.7 MiB
docker-ce-cli x86_64 1:29.1.3-1.fc43 docker-ce-stable 34.1 MiB
docker-compose-plugin x86_64 5.0.0-1.fc43 docker-ce-stable 29.9 MiB
Installing dependencies:
libcgroup x86_64 3.0-9.fc43 fedora 153.7 KiB
Installing weak dependencies:
docker-ce-rootless-extras x86_64 29.1.3-1.fc43 docker-ce-stable 11.3 MiB

Transaction Summary:
Installing: 7 packages

Total size of inbound packages is 94 MiB. Need to download 94 MiB.
After this operation, 367 MiB extra will be used (install 367 MiB, remove 0 B).
Is this ok [y/N]: Y
[1/7] docker-ce-cli-1:29.1.3-1.fc43.x86_64 100% | 847.1 KiB/s | 8.3 MiB | 00m10s
[2/7] docker-ce-3:29.1.3-1.fc43.x86_64 55% [=====] | 1.4 MiB/s | 12.0 MiB | 00m06s
[3/7] containerd.io-0:2.2.1-1.fc43.x86_64 29% [=====] | 1.2 MiB/s | 10.3 MiB | 00m20s
[4/7] docker-buildx-plugin-0:0.30.1-1.fc43.x86_64 0% [<=] | 1.0 B/s | 0.0 B | ?
-----
[1/7] Total 32% [=====] | 3.4 MiB/s | 30.6 MiB | 00m18s
```

2.2 Aktivasi dan Konfigurasi Docker Service

Setelah instalasi selesai, Docker service diaktifkan menggunakan perintah `systemctl enable --now docker`. Perintah ini berfungsi untuk menjalankan Docker daemon secara langsung serta memastikan Docker otomatis berjalan saat sistem dinyalakan. Aktivasi service ini merupakan langkah penting agar Docker dapat digunakan untuk menjalankan container.

Selain itu, dilakukan konfigurasi hak akses pengguna dengan menambahkan user ke dalam grup docker menggunakan perintah `usermod -aG docker $USER`. Konfigurasi ini bertujuan agar pengguna dapat menjalankan perintah Docker tanpa perlu menggunakan hak akses superuser (`sudo`).

```
[1/7] docker-ce-cli-1:29.1.3-1.fc43.x86_64 100% | 612.1 KiB/s | 8.3 MiB | 00m14s
[2/7] containerd.io-0:2.2.1-1.fc43.x86_64 100% | 1.3 MiB/s | 35.3 MiB | 00m28s
[3/7] docker-ce-3:29.1.3-1.fc43.x86_64 100% | 670.1 KiB/s | 21.7 MiB | 00m33s
[4/7] libcgrou-0:3.0-9.fc43.x86_64 100% | 81.4 KiB/s | 73.6 KiB | 00m01s
[5/7] docker-compose-plugin-0:5.0.0-1.fc43.x86_64 100% | 1.1 MiB/s | 8.0 MiB | 00m07s
[6/7] docker-ce-rootless-extras-0:29.1.3-1.fc43.x86_64 100% | 1.2 MiB/s | 3.4 MiB | 00m03s
[7/7] docker-buildx-plugin-0:0.30.1-1.fc43.x86_64 100% | 747.7 KiB/s | 17.0 MiB | 00m23s
-----
[7/7] Total 100% | 2.5 MiB/s | 93.9 MiB | 00m38s
Running transaction
[1/9] Verify package files 100% | 8.0 B/s | 7.0 B | 00m01s
[2/9] Prepare transaction 100% | 15.0 B/s | 7.0 B | 00m00s
[3/9] Installing libcgrou-0:3.0-9.fc43.x86_64 100% | 11.7 MiB/s | 155.1 KiB | 00m00s
[4/9] Installing containerd.io-0:2.2.1-1.fc43.x86_64 100% | 176.5 MiB/s | 120.0 MiB | 00m01s
[5/9] Installing docker-ce-cli-1:29.1.3-1.fc43.x86_64 100% | 144.8 MiB/s | 34.2 MiB | 00m00s
[6/9] Installing docker-ce-3:29.1.3-1.fc43.x86_64 100% | 184.7 MiB/s | 93.7 MiB | 00m01s
[7/9] Installing docker-ce-rootless-extras-0:29.1.3-1.f 100% | 128.2 MiB/s | 11.3 MiB | 00m00s
[8/9] Installing docker-compose-plugin-0:5.0.0-1.fc43.x 100% | 164.0 MiB/s | 29.9 MiB | 00m00s
[9/9] Installing docker-buildx-plugin-0:0.30.1-1.fc43.x 100% | 42.2 MiB/s | 77.9 MiB | 00m02s
Complete!

13:20:24 akbarrzk@fedora ~
$ sudo systemctl enable --now docker
Created symlink '/etc/systemd/system/multi-user.target.wants/docker.service' -> '/usr/lib/systemd/system/docker.service'.

13:20:49 akbarrzk@fedora ~
$ sudo usermod -aG docker $USER
```

4.3 Pengujian Instalasi Docker

Untuk memastikan bahwa Docker telah terinstal dan berjalan dengan baik, dilakukan pengujian menggunakan perintah `docker run hello-world`. Pada percobaan awal, sistem menampilkan pesan `permission denied` karena sesi terminal belum memperbarui hak akses grup pengguna. Setelah memperbarui sesi menggunakan perintah `newgrp docker`, pengujian kembali dilakukan.

Hasil pengujian menunjukkan bahwa Docker berhasil menarik (pull) image `hello-world` dari Docker Hub dan menjalankan container dengan sukses. Pesan “Hello from Docker!” menandakan bahwa Docker daemon, client, dan container berjalan dengan baik serta instalasi Docker telah berhasil.

```
13:20:59 akbarrzk@fedora ~  
$ docker run hello-world  
permission denied while trying to connect to the docker API at unix:///var/run/docker.sock  
  
13:21:04 akbarrzk@fedora ~  
$ newgrp docker  
  
13:21:52 akbarrzk@fedora ~  
$ docker run hello-world  
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
17eec7bbc9d7: Pull complete  
ea52d2000f90: Download complete  
Digest: sha256:d4aaab6242e0cace87e2ec17a2ed3d779d18fbfd03042ea58f2995626396a274  
Status: Downloaded newer image for hello-world:latest  
  
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
  
To generate this message, Docker took the following steps:  
1. The Docker client contacted the Docker daemon.  
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
   (amd64)  
3. The Docker daemon created a new container from that image which runs the  
   executable that produces the output you are currently reading.  
4. The Docker daemon streamed that output to the Docker client, which sent it  
   to your terminal.  
  
To try something more ambitious, you can run an Ubuntu container with:  
$ docker run -it ubuntu bash  
  
Share images, automate workflows, and more with a free Docker ID:  
https://hub.docker.com/  
  
For more examples and ideas, visit:  
https://docs.docker.com/get-started/  
  
13:22:02 akbarrzk@fedora ~  
$
```

4.4 Analisis hasil Instalasi

Berdasarkan hasil instalasi dan pengujian yang telah dilakukan, dapat disimpulkan bahwa Docker berhasil diinstal dan dikonfigurasi dengan baik pada sistem. Seluruh komponen utama Docker berjalan sesuai fungsinya, mulai dari Docker daemon, image management, hingga eksekusi container sederhana.

Instalasi Docker ini menjadi dasar penting untuk tahap selanjutnya dalam proyek, seperti pembuatan Dockerfile, Docker image, serta pengelolaan container dan resource sistem. Selain itu, proses ini memberikan pemahaman awal mengenai peran sistem operasi dalam mendukung teknologi container melalui mekanisme kernel Linux.

BAB III

PENUTUP

3.1 Kesimpulan

Berdasarkan progres yang telah dilakukan, dapat disimpulkan bahwa proses instalasi Docker pada sistem operasi Fedora Linux berhasil dilaksanakan dengan baik. Seluruh tahapan instalasi, mulai dari penambahan repository resmi Docker, pemasangan paket Docker beserta dependensinya, hingga aktivasi Docker service, dapat berjalan tanpa kendala yang berarti.

Hasil pengujian menggunakan perintah `docker run hello-world` menunjukkan bahwa Docker daemon dan client berfungsi dengan 정상, serta mampu menjalankan container sesuai dengan yang diharapkan. Pesan keberhasilan yang ditampilkan menandakan bahwa lingkungan Docker telah siap digunakan untuk menjalankan aplikasi berbasis container.

Dengan selesainya progres instalasi ini, sistem telah memiliki lingkungan container yang siap digunakan sebagai dasar untuk tahapan selanjutnya, seperti pembuatan Dockerfile, pembuatan Docker image, serta pengelolaan container dan resource sistem. Progres ini juga memberikan pemahaman awal mengenai keterkaitan antara sistem operasi dan teknologi container dalam mendukung virtualisasi ringan.

3.2 Bukti Kerja

