IMPLEMENTASI DOCKER UNTUK CLOUD COMPUTING PADA FEDORA LINUX: STUDI KASUS PYTHON DALAM KONTAINER

Lastri Elisabet Butarbutar[1], Sherly Davina[2], Vivielda Farmawaty Tambunan[3]
elisabetlastri12@gmail.com[1], sherlydavinaa@gmail.com[2] ,
vivieldafarmawatytambunan@gmail.com[3]
Universitas Negeri Medan

ABSTRAK

Penelitian ini bertujuan untuk menjelaskan implementasi Docker pada sistem operasi Fedora Linux sebagai lingkungan pengembangan dan pengelolaan aplikasi Python dalam container. Docker memungkinkan aplikasi berjalan secara konsisten di berbagai lingkungan dengan mengisolasi aplikasi beserta seluruh dependensinya. Penelitian ini menggunakan pendekatan studi kasus dengan beberapa tahapan, meliputi instalasi Docker, pembuatan Dockerfile, hingga tahap pengunggahan image aplikasi ke Docker Hub. Hasil implementasi menunjukkan bahwa Docker di Fedora Linux mampu menyediakan lingkungan yang efisien dan mudah dikonfigurasi untuk menjalankan aplikasi Python. Hasil ini memberikan gambaran tentang fleksibilitas penggunaan Docker dalam pengelolaan aplikasi yang terisolasi, tanpa ketergantungan pada platform tertentu.
Kata Kunci: Docker, Fedora Linux, Python, Kontainer, Docker Hub

ABSTRACT

This research aims to explain the implementation of Docker on Fedora Linux as a development and management environment for Python applications within containers. Docker allows applications to run consistently across various environments by isolating applications along with their dependencies. This study employs a case study approach through several stages, including Docker installation, Dockerfile creation, and application image upload to Docker Hub. The implementation results demonstrate that Docker on Fedora Linux provides an efficient and easily configurable environment for running Python applications. These findings illustrate Docker's flexibility in managing isolated applications without dependency on a specific platform.
Keywords*: Docker, Fedora Linux, Python, Container, Docker Hub*

PENDAHULUAN

Dalam era kemajuan teknologi yang terus berkembang, aplikasi berbasis web telah menjadi salah satu elemen terpenting dalam ekosistem digital global. Aplikasi web saat ini memainkan peran utama dalam berbagai bidang, termasuk bisnis, pendidikan, hiburan, dan interaksi sosial. Keberhasilan aplikasi web memberikan dampak positif yang signifikan dalam hal komunikasi, konektivitas global, dan efisiensi operasional. Namun, di balik kesuksesan tersebut, terdapat sejumlah tantangan yang perlu diatasi yang mempengaruhi efisiensi, performa, dan ketersediaan aplikasi berbasis web. Salah satu permasalahan yang mendominasi adalah performa aplikasi web yang kurang optimal. Seringnya, pengguna menghadapi pengalaman yang kurang memuaskan, seperti waktu pemuatan halaman yang tinggi atau respon server yang lambat (Ekaputra, 2023).

Cloud computing telah menjadi salah satu solusi utama dalam memenuhi kebutuhan komputasi modern, yang menuntut skalabilitas, fleksibilitas, dan efisiensi biaya. Seiring dengan meningkatnya permintaan terhadap layanan cloud, teknologi kontainerisasi seperti Docker mulai memainkan peran penting dalam menghadirkan lingkungan yang mudah dikonfigurasi, ringan, dan dapat diandalkan.

Docker memungkinkan aplikasi untuk berjalan secara konsisten di berbagai lingkungan dengan cara mengemasnya beserta seluruh dependensi yang diperlukan.

Docker adalah suatu platform terbuka bagi pengembang perangkat lunak dan pengelola sistem jaringan untuk membangun, mengirimkan dan menjalankan aplikasi-aplikasi terdistribusi. Definisi tersebut membawa pengertian praktis bahwa Docker merupakan suatu cara memasukkan layanan ke dalam lingkungan terisolasi bernama container, sehingga layanan tersebut dapat dipaketkan menjadi satu bersama dengan pustaka dan software lain yang dibutuhkan (Imammuddin,2019).

Dalam pengembangan aplikasi, pengembang biasanya menggunakan virtualisasi pada server sehingga proses pembuatan program dapat berjalan pada berbagai platform dan konfigurasi hardware. Masalah yang dihadapi dengan virtualisasi adalah perlunya menyiapkan satu sistem operasi secara utuh, termasuk berbagai aplikasi yang dibawa sistem tersebut. Bisa dibayangkan dengan banyaknya virtualisasi yang berjalan di sebuah server akan memberatkan sistem tersebut. Docker continer adalah teknik dimana mengisolasi suatu sistem sehingga tidak mengganggu sistem yang lainnya. Docker menggunakan virtualisasi berbasis linux dan container, yang mengisolasikan proses tanpa mengisolasi perangkat keras seperti kernel atau system operasi sehingga dapat mengurangi overhead yang lebih rendah pada hardware.

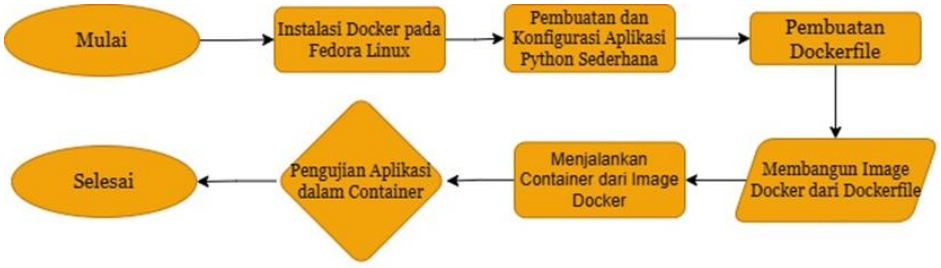
Dengan diterapkannya sistem virtualisasi server berbasis Docker container, diharapkan dapat meningkatkan kinerja sebuah server dan memudahkan proses deployment (penyebaran) aplikasi web beserta software pendukung seperti webserver, database server, dll ke server. Penelitian ini membahas bagaimana penerapan Docker pada konteks cloud computing pada Fedora Linux.

METODE PENELITIAN

1. Jenis Penelitian

Penelitian ini menggunakan metode eksperimen dengan pendekatan studi kasus. Eksperimen dilakukan untuk menguji implementasi Docker pada Fedora Linux dan melihat bagaimana aplikasi Python dapat dijalankan dalam container.

2. Tahapan dan Prosedur Penelitian



3. Alat dan Bahan:

Penelitian ini menggunakan beberapa alat dan bahan utama untuk mendukung eksperimen dan implementasi Docker pada Fedora Linux. Berikut daftar alat dan bahan yang digunakan:

- Fedora Linux ssebagai sistem operasi utama.
- Docker sebagai platform cntainerization
- Aplikasi Python (misalnya, server web menggunakan Flask)
- Komputer
- Docker Hub untuk menditribusikan image yang ditarik dan di-build.

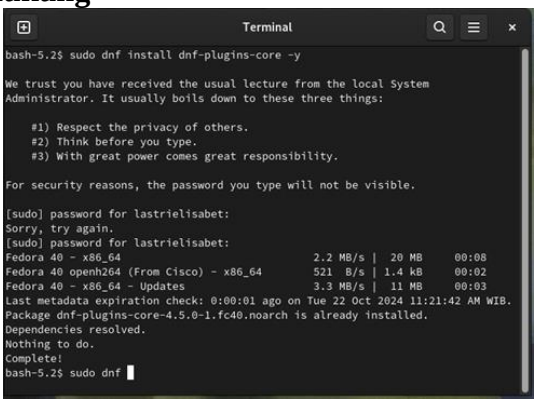
HASIL DAN PEMBAHASAN

A. Hasil Implementasi Docker pada Fedora Linux

Pada penelitian ini, fokus utama adalah menjelaskan bagaimana Docker diimplementasikan pada sistem operasi Fedora Linux, serta bagaimana Docker digunakan

untuk menjalankan aplikasi Python dalam container. Implementasi ini mencakup beberapa tahapan mulai dari instalasi Docker, penyiapan Dockerfile, hingga proses pull image dari Docker hub. Berikut adalah hasil dari tiap tahapan tersebut

1. Instalasi Paket Pendukung



1 Instalasi Paket Pendukung

Sebelum menginstal Docker, Pastikan anda menyiapkan paket-paket pendukung yang diperlukan

2. Menambahkan Repository Docker

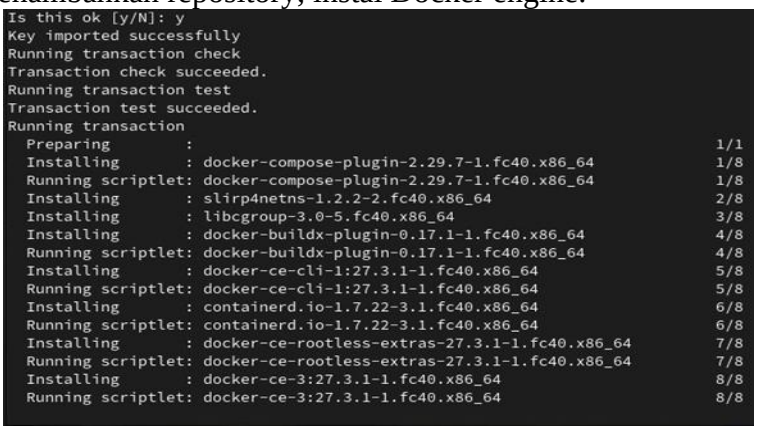
2 Menambahkan Repository Docker

Untuk mengakses versi docker terbaru, tambahkan repository docker.

3. Instalasi Docker Engine

3 Instalasi Docker Engine

Setelah menambahkan repository, instal Docker engine.



```
Installed:
  containerd.io-1.7.22-3.1.fc40.x86_64
  docker-buildx-plugin-0.17.1-1.fc40.x86_64
  docker-ce-3:27.3.1-1.fc40.x86_64
  docker-ce-cli-1:27.3.1-1.fc40.x86_64
  docker-ce-rootless-extras-27.3.1-1.fc40.x86_64
  docker-compose-plugin-2.29.7-1.fc40.x86_64
  libcgroup-3.0-5.fc40.x86_64
  slirp4netns-1.2.2-2.fc40.x86_64

Complete!
```

Ini adalah Docker yang sudah terinstal.

4. Memulai dan Aktifkan Layanan Docker

```
bash-5.2$ sudo systemctl start docker
[sudo] password for lastrielisabet:
bash-5.2$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /us
r/lib/systemd/system/docker.service.
bash-5.2$
```

4 Memulai dan Aktifkan Layanan Docker

Setelah instalasi selesai, mulai dan aktifkan layanan Docker.

5. Verifikasi Instalasi

```
bash-5.2$ sudo docker --version
Docker version 27.3.1, build ce12230
bash-5.2$
```

5 Verifikasi Instalasi

Untuk memastikan Docker sudah terinstal dan berjalan.

6. Menambahkan Pengguna ke Grup Docker

```
bash-5.2$ sudo docker ps
[sudo] password for lastrielisabet:
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
bash-5.2$ sudo usermod -aG docker $(whoami)
bash-5.2$ newgrp docker
bash-5.2$
```

6 Menambahkan Pengguna ke Grup Docker

Perintah sudo docker ps diatas menjalankan perintah Docker dengan menggunakan sudo. Pada Langkah berikutnya, kita akan menambahkan pengguna ke dalam grup Docker agar dapat dijalankan tanpa harus menggunakan perintah sudo setiap kali.

```
bash-5.2$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
bash-5.2$
```

Setelah dicoba lagi tidak menggunakan sudo, ternyata sudah bisa.

7. Memulai Pengunaan Docker pada Cloud Computing

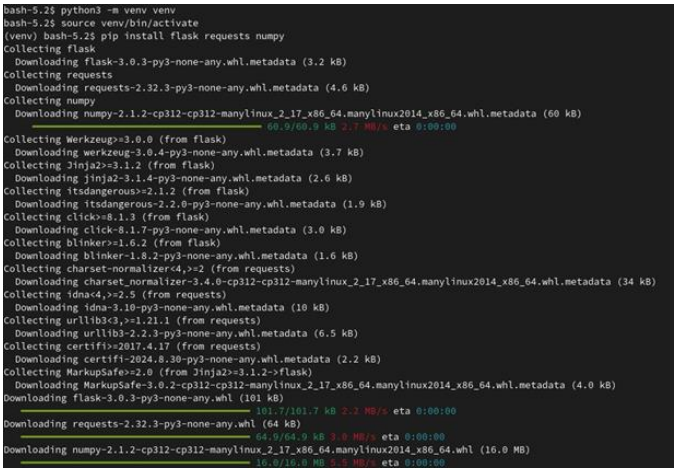
Setelah Docker terinstal, kita dapat mulai menggunakannya untuk kebutuhan cloud computing, termasuk membangun dan menjalankan container yang akan digunakan dalam lingkungan cloud. sebelum aplikasi dapat digunakan di cloud, kita perlu membuat Dockerfile yang berisi instruksi untuk membuat image dari aplikasi yang akan di-deploy. Sebagai contoh, kita dapat membuat Dockerfile untuk aplikasi berbasis Python. Namun, sebelum itu pastikan file requirements.txt sudah tersedia di direktori kerja. File ini dapat dihasilkan secara otomatis dengan menggunakan virtual environment dan menginstal dependensi melalui pip.

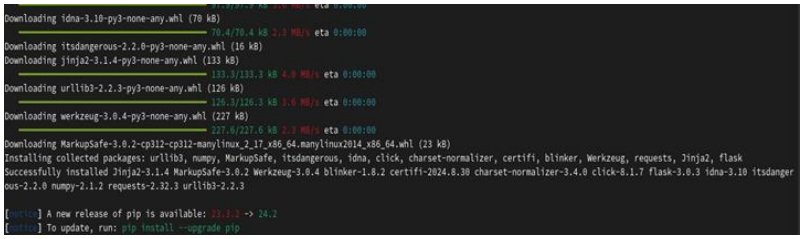
Membuat virtual environment

```
bash-5.2$ python3 -m venv venv
bash-5.2$ source venv/bin/activate
```

7 Memulai Penggunaan Docker Pada Cloud Computing

Setelah berhasil, lingkungan virtual akan aktif, dan tampilan anda akan berubah seperti pada gambar berikut.





8. Menghasilkan Requirements.txt

```
(venv) bash-5.2$ pip freeze > requirements.txt
```

8 Menghasilkan Requirements.txt

Setelah semua dependensi terpasang, kita dapat menghasilkan file requirements.txt

9. Verifikasi Isi File

```
(venv) bash-5.2$ cat requirements.txt
blinker==1.8.2
certifi==2024.8.30
charset-normalizer==3.4.0
click==8.1.7
Flask==3.0.3
idna==3.10
itsdangerous==2.2.0
Jinja2==3.1.4
MarkupSafe==3.0.2
numpy==2.1.2
requests==2.32.3
urllib3==2.2.3
Werkzeug==3.0.4
```

9 Verifikasi Isi File

Setelah requirements.txt dibuat, kita bisa melihat isinya dengan perintah ini, maka akan ditampilkan daftar dependensi dan versi yang kita gunakan.

10. Menginstal Dependensi Menggunakan requirements.txt di Dockerfile

Sekarang, Anda dapat menggunakan file requirements.txt ini di dalam Dockerfile untuk menginstal semua dependensi dengan perintah: RUN pip install -r requirements.txt

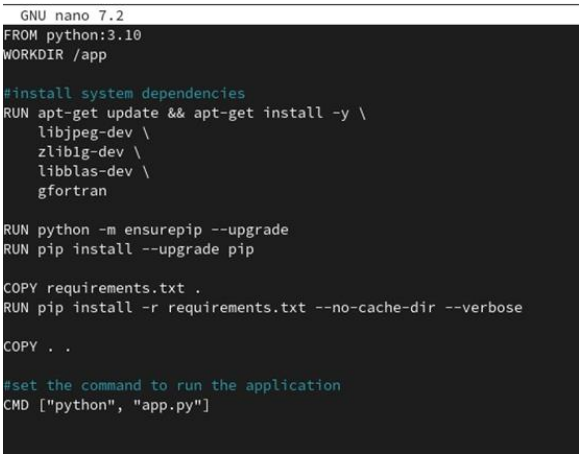
11. Membuat Dockerfile dengan Perintah Nano

```
bash-5.2$ nano Dockerfile
```

10 Membuat Dockerfile dengan Perintah Nano

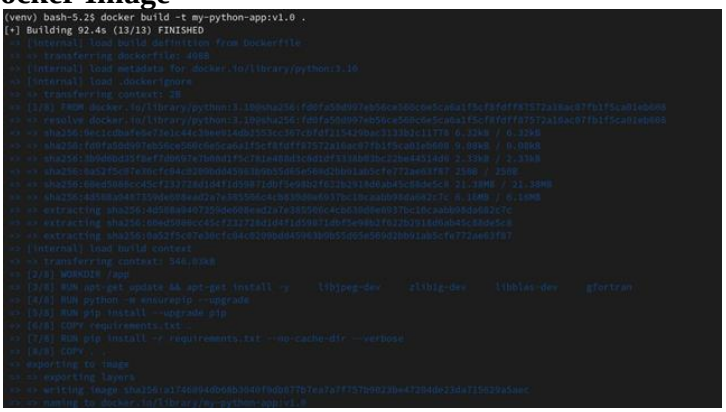
Perintah nano Dockerfile digunakan untuk membuka atau membuat file bernama Dockerfile menggunakan editor teks nano di terminal Linux/Unix

12. Isi Nano Dockerfile



11 Isi Nano Dockerfile

13. Membuat Docker Image



12 Membuat Docker Image

Proses ini menghasilkan Docker image yang dapat digunakan untuk menjalankan aplikasi di dalam container.

Docker image berfungsi untuk menyatukan semua dendensi aplikasi. Image yang dihasilkan dari proses build mencakup semua komponen yang dibutuhkan aplikasi agar bisa berjalan tanpa bergantung pada sistem operasi atau konfigurasi perangkat yang menjalankannya. Dengan demikian, aplikasi dapat dijalankan di mana saja, asalkan ada Docker—tanpa harus khawatir tentang konflik dependensi atau konfigurasi yang berbeda. Docker image yang dihasilkan dari proses build berisi semua yang dibutuhkan aplikasi agar bisa berjalan tanpa bergantung pada sistem operasi atau konfigurasi perangkat yang menjalankannya. Ini berarti kamu bisa menjalankan aplikasi di mana saja, asalkan ada Docker—tanpa harus khawatir tentang konflik dependensi atau konfigurasi yang berbeda.

14. Integrasi dengan Cloud Computing

Beberapa langkah tambahan yang diperlukan untuk melakukan deployment aplikasi berbasis container ke cloud. Namun, sebelum itu, perlu dilakukan pembuatan akun di Docker hub.

1) Mengunggah image ke Docker Hub

Setelah aplikasi dikemas ke dalam docker image, langkah selanjutnya adalah mengunggah image tersebut ke registry container, seperti Docker Hub, agar dapat diakses oleh platform cloud.

Setelah membuat akun, kita melakukan login

```
(venv) bash-5.2$ docker login

USING WEB-BASED LOGIN
To sign in with credentials on the command line, use 'docker login -u <username>'

Your one-time device confirmation code is: LSXW-JXCL
Press ENTER to open your browser or submit your device code here: https://login.docker.com/activate

Waiting for authentication in the browser…
docker login -u Lastributarbutar7
WARNING! Your password will be stored unencrypted in /home/lastrielisabet/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded
```

Tag Image

```
(venv) bash-5.2$ docker tag my-python-app:v1.0 dockerhub_username/my-python-app:v1.0
```

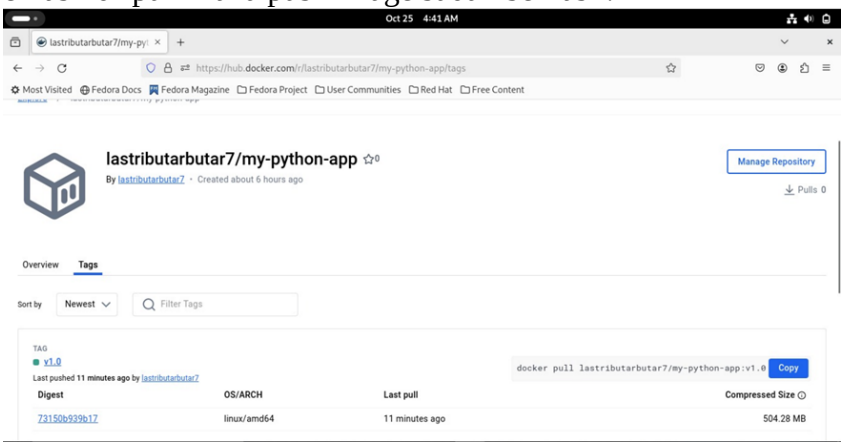
Push ke docker hub

```
Terminal
bash-5.2$ docker push lastributarbutar7/my-python-app:v1.0
The push refers to repository [docker.io/lastributarbutar7/my-python-app]
24dab1a9b6eb: Pushed
e69338deb92c: Layer already exists
798603d744f2: Layer already exists
3729ba9f7782: Layer already exists
515548c15cce: Layer already exists
98fc4d2ef9f4: Layer already exists
12eba4a133b0: Layer already exists
70f3c3d676be: Layer already exists
9120c003805e: Layer already exists
d8c424527cde: Layer already exists
d23b5e6144a7: Pushed
e5ee1bd83fe3: Pushed
43da071b5e0c: Layer already exists
ef5f5ddeb0a6: Pushed
v1.0: digest: sha256:73150b939b170d572ff87217f4651968fb4fb5bfe1041ad8efb0e057d51
6d75a size: 3267
```

Langkah Verifikasi

```
bash-5.2$ docker pull lastributarbutar7/my-python-app:v1.0
v1.0: Pulling from lastributarbutar7/my-python-app
Digest: sha256:73150b939b170d572ff87217f4651968fb4fb5bfe1041ad8efb0e057d516d75a
Status: Image is up to date for lastributarbutar7/my-python-app:v1.0
docker.io/lastributarbutar7/my-python-app:v1.0
bash-5.2$
```

Melakukan pull dari Docker Hub untuk memastikan image sudah tersedia, jika sudah berhasil di-pull maka push image sudah berhasil.



```
bash-5.2$ docker images
REPOSITORY                                TAG       IMAGE ID       CREATED
     SIZE
Lastributarbutar7/my-python-app           v1.0      9e192a6e890d   13 hours ag
o    1.38GB
lastributarbutar7/my-python-app           v1.0      9e192a6e890d   13 hours ag
o    1.38GB
my-python-app                             v1.0      9e192a6e890d   13 hours ag
o    1.38GB
Lastributarbutar7-dockerhub/my-python-app v1.0      be817bf3e7ac   24 hours ag
o    1.38GB
dockerhub_username/my-python-app          v1.0      a1746094db68   26 hours ag
o    1.28GB
php                                       apache    e29e62f1e0c5   8 days ago
     507MB
```

Penelitian ini mengevaluasi implementasi Docker pada Fedora Linux dengan tujuan memudahkan pengelolaan aplikasi berbasis Python dalam container. Metode eksperimen digunakan untuk menguji proses instalasi dan konfigurasi Docker. Tahapan penelitian dimulai dengan instalasi Docker di Fedora versi 40, termasuk memastikan ketersediaan paket pendukung, menambahkan repository Docker, dan menginstal Docker Engine menggunakan perintah dnf. Verifikasi instalasi dilakukan dengan perintah sudo docker ps, dan untuk mempermudah penggunaan, pengguna ditambahkan ke grup Docker agar tidak perlu menggunakan perintah sudo setiap kali. Setelah Docker terinstal, Dockerfile dibuat untuk membangun Docker image, yang berisi instruksi pembuatan lingkungan aplikasi. File requirements.txt dihasilkan secara otomatis melalui virtual environment untuk mencatat semua dependensi yang dibutuhkan, dan perintah RUN pip install -r requirements.txt digunakan dalam Dockerfile untuk memasang semua dependensi.

Keuntungan menggunakan Docker adalah kemampuannya untuk menjalankan aplikasi di berbagai lingkungan tanpa bergantung pada sistem operasi tertentu, sehingga mengurangi konflik dependensi. Setelah Docker image siap, kami mengunggahnya ke Docker Hub setelah memberi tag, memungkinkan aplikasi untuk di-deploy di cloud dengan mudah. Secara keseluruhan, penelitian ini menunjukkan bahwa Docker adalah solusi yang efisien dan portabel untuk pengelolaan aplikasi berbasis Python pada Fedora Linux.

KESIMPULAN

Penelitian ini menunjukkan bahwa implementasi Docker pada Fedora Linux memberikan kemudahan dalam mengelola dan menjalankan aplikasi Python dalam container yang terisolasi. Dengan menggunakan Docker, aplikasi dapat dikemas bersama seluruh dependensi yang dibutuhkan, sehingga meminimalkan konflik yang biasanya muncul akibat perbedaan lingkungan. Docker Hub juga berperan penting dalam mendistribusikan image, memungkinkan pengguna lain untuk mengunduh dan menjalankan aplikasi secara konsisten di berbagai platform. Secara keseluruhan, Docker di Fedora Linux terbukti menjadi solusi yang efisien, portabel, dan mudah dikonfigurasi untuk pengelolaan aplikasi berbasis Python, mendukung fleksibilitas dan konsistensi dalam pengembangan perangkat lunak modern

DAFTAR PUSTAKA

Ariadi, F., Iswahyudi, C., Nurnawati, E., Informatika, J., & Akprind, I. (2020). Penerapan Docker Container Sebagai Teknologi Ramah Skalabilitas Dibanding Teknik Virtualisasi Untuk Membangun Website Di Ubuntu 18.04.4 Lts. Jarkom, 8(2), 47–57.

Dame, A. H., Zailani, A. U., Kunci, K., Implementasi, :, & Server, W. (2023). Implementasi Webserver Berbasis Docker Dan Linux. Jurnal Ilmu Komputer Dan Pendidikan, 1(5), 1084–1090.

Dwiyatno, S., Rachmat, E., Sari, A. P., & Gustiawan, O. (2020). Implementasi Virtualisasi Server

Berbasis Docker Container. PROSISKO: Jurnal Pengembangan Riset Dan Observasi Sistem Komputer, 7(2), 165–175.

Ekaputra, A. R., & Affandi, A. S. (2023). Pemanfaatan layanan cloud computing dan docker container untuk meningkatkan kinerja aplikasi web. Journal of Information System and Application Development, 1(2), 138–147.

Felani, R., Al Azam, M. N., Adi, D. P., Widodo, A., & Gumelar, A. B. (2020). Optimizing Virtual Resources Management Using Docker on Cloud Applications. IJCCS (Indonesian Journal of Computing and Cybernetics Systems), 14(3), 319.

Imammuddin, Januar Al-Amien, M. K. (2019). Membangun Cloud Menggunakan Docker Pada Implementasi Load Balancing dan Pengujian Algoritma Round Robin Pada Web Server. Prosiding Seminar Nasional Computation Technology and Its Aplication, 1(1), 17–21.

Megantara, R. A., Alzami, F., Pramunendar, R. A., & Prabowo, D. P. (2022). Pengembangan Dan Implementasi Docker Untuk Memaksimalkan Utilitas Server Universitas Pada Masa Covid-19. Transmisi, 24(2), 48–54.

Mukmin, C., Naraloka, T., & Andriyanto, Q. H. (2021). ANALISIS PERBANDINGAN KINERJA LAYANAN CONTAINER AS A SERVICE (CAAS) Studi Kasus : Docker dan Podman. Kumpulan JurnaL Ilmu Komputer (KLIK), 08(2), 152–161.

Umar, I. A., Nurhadi, N., Syafaah, L., & Khaeruddin, K. (2021). Implementasi Sistem Aplikasi Docker Terintegrasi Openstack. Jurnal Informatika Dan Rekayasa Elektronik, 4(1), 60–67.