Journal of Mechatronic and Electrical Engineering
Homepage: http://ejournal.umm.ac.id/index.php/jmee
ISSN: XXXX-XXXX
Vol. 1, No. 1, April 2021, pp: 42-51

Implementasi CDN(*Content Delivery Network*) Menggunakan *Cloudflare* terintegrasi Dengan *Docker Container*

Haikal Alham Tuara#[1] , NurAlif Maridyah[2], Khaerudin Khaerudin[3]

Info Artikel dan Penulis

1,2,3 - Program Studi Teknik Elektro, Universitas Muhammadiyah Malang, Jl. Tlogomas No. 246, Malang, Indonesia

Penulis korespondensi :
ekaltuara@gmail.com

Kata Kunci:
CDN(*Content Delivery Network*), *Docker Container*

Proses Artikel:
Dikirim 27 Maret 2021
Direvisi 30 Maret 2021
Diterima 31 Maret 2021
Diterbitkan online 22 April 2021

Abstrak

Teknologi CDN dapat menyediakan lebih dari 1 server (multi server) dan beberapa fungsi lainya yang sangat diperlukan user untuk mengakses server, salah satu yang diperlukan user dalam melakukan akses terhadap server adalah ketika server tersebut mampu menyediakan bandwidth yang cukup untuk mendukung permintaan user, sehingga .

Teknologi CDN dapat menyediakan lebih dari 1 server (multi server) dan beberapa fungsi lainya yang sangat diperlukan user untuk mengakses server, Salah satu yang diperlukan user dalam melakukan akses terhadap server adalah ketika server tersebut mampu menyediakan bandwidth yang cukup untuk mendukung permintaan user. sehingga ketika user mengakses sebuah website kemungkinan terjadinya overload lebih kecil. Melalui teknologi CDN, client tidak membutuhkan waktu yang lama untuk mengakses website yang mempunyai hosting di Australia, ketika website tersebut mempunyai server pengganti di jepang atau di negara yang letaknya berdekatan dengan jepang. Docker Container Merupakan sebuah program yang dapat membantu untuk mengembangkan dan menjalankan aplikasi dalam system operasi manapun, docker menyediakan sebuah wadah khusus yang disebut container, container berisi segala sesuatu yang diperlukan aplikasi agar dapat berjalan dengan baik di system operasi apapun baik windows,linux, dan mac os, docker juga berfungsi sebagai penyedia layanan virtual bagi container yang akan berjalan di system operasi induk atau host , docker menyediakan beberapa sumber daya yang dibutuhkan oleh container seperti akses file, jaringan internet dan port agar aplikasi dapat berjalan dengan dengan sempurna.

Abstract

CDN technology can provide more than 1 server (multi server) and several other functions that are needed by users to access the server, One of the things that aneeds user to access the server is when the server is able to provide bandwidth sufficientto supportrequests user. so that when a user accesses a website the possibility of overloading is smaller. Through CDN technology, clients do not need a long time to access a website that has hosting in Australia, when the website has aserver replacementin Japan or in a country that is located close to Japan. Docker Container is a program that can help to develop and run applications in any operating system, docker provides a special container called a container, a container that contains everything an application needs to run well on any operating system, Windows, Linux, and Mac. OS, docker also functions as a virtual service provider for containers that will run on the parent or host operating system, Docker provides some of the resources needed by containers such as file access, internet networks and ports so that applications can run perfectly.

Tuara, H. A. T., Maridyah, N. A. ., & Khaerudin, K. (2021). Implementasi CDN(Content Delivery Network) menggunakan Cloudflare terintegrasi dengan Docker Countainer. Journal of Mechatronic and Electrical Engineering, Vol. 1(1), pp: 42-51, April 2021, doi: https://doi.org/10.22219/jmee.xxxx.xxxx

1. PENDAHULUAN

Kemajuan teknologi khususnya internet di berbagai belahan dunia telah berkembang dan mengalami peningkatan yang cukup sangat pesat. Internet pada saat ini sangat mempunyai manfaat yang sangat besar untuk mencari berbagai sumber informasi dan juga digunakan untuk berkomunikasi, hal ini menyebabkan suatu *website* membutuhkan *bandwitch* yang sangat besar. Akibatnya *server* dari *website* tersebut mengalami *overload* dan kualitas layanan dapat menurun[1]. Dengan demikian, diperlukan solusi untuk meningkatan kualitas *server* sehinggan

kebutuhan *user* dapat diimbangi oleh layanan *server*. Dibutuhkan sebuah system yang mampu menjawab permasalahan dari *user*.

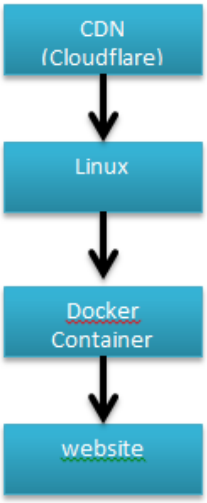
Teknologi CDN dapat menyediakan lebih dari 1 *server* (*multi server*) dan beberapa fungsi lainya yang sangat diperlukan *user* untuk mengakses *server*, Salah satu yang diperlukan *user* dalam melakukan akses terhadap *server* adalah ketika *server* tersebut mampu menyediakan *bandwidth* yang cukup untuk mendukung permintaan *user*. sehingga ketika *user* mengakses sebuah *website* kemungkinan terjadinya *overload* lebih kecil. Melalui teknologi CDN, *client* tidak membutuhkan waktu yang lama untuk mengakses *website* yang mempunyai hosting di Australia, ketika *website* tersebut mempunyai *server* pengganti di jepang atau di negara yang letaknya berdekatan dengan jepang.[2]

Docker Container Merupakan sebuah program yang dapat membantu untuk mengembangkan dan menjalankan aplikasi dalam system operasi manapun, docker menyediakan sebuah wadah khusus yang disebut container, container berisi segala sesuatu yang diperlukan aplikasi agar dapat berjalan dengan baik di system operasi apapun baik windows,linux, dan mac os, docker juga berfungsi sebagai penyedia layanan virtual bagi container yang akan berjalan di system operasi induk atau host , docker menyediakan beberapa sumber daya yang dibutuhkan oleh container seperti akses file, jaringan internet dan port agar aplikasi dapat berjalan dengan dengan sempurna.[3]

2. METODE

2.1 Perancangan system CDN (*Content Delivery Network*) menggunakan Cloudflare yang terinterasi dengan Docker Container.

Dalam tahap perancangan ini ada beberapa komponen yang sangat dibutuhkan yaitu, Linux berfungsi sebagai sistem operasi, Docker Container sebagai penyedia layanan virtual bagi container yang akan berjalan di sistem operasi induk atau host, portainer berfungsi untuk mengelola kontainer, CDN (Cloudflare) berfungsi untuk menambah kinerja dari suatu website. Gambar 1 dibawah ini adalah gambaran blok diagram system.

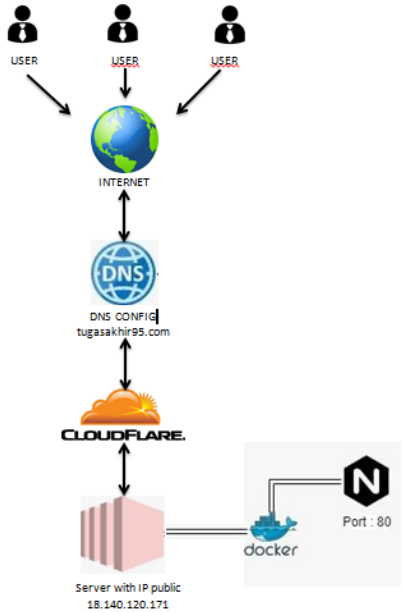


Gambar 1. Diagram Blok Perancangan System

Berdasarkan blok diagram pada gambar 1 dapat dilihat bahwa tahap awal yang harus dilakukan adalah menkonfigurasi CDN menggunakan Cloudflare, selanjutnya adalah menginstal Docker Container di Linux, setelah tahapan penginstalan Docker Container pada Linux berhasil, kemudian mengintegrasikan CDN menggunakan Cloudflare dengan Docker Container.

2.2 Topologi Sistem

Agar memperkirakan bayangan bagaimana cara kerja sistem tersebut, langkah yang harus dilakukan selanjutnya membuat pemodelan topologi system, berikut ini merupakan desain topologi sistem dari cara bagaimana *user* mengakses *website* yang sudah diimplemntasi *CDN* dan *Docker Container* telah terintegrasi. Pada gambar 2 menjelaskan desain topologi sistem yang terdapat beberapa *user*, dimana beberapa *user* tersebut mengakses internet yang berdomain tugasakhir95.com sebagai halaman *Website*, dimana pada halaman tersebut sudah diimplementasikan Cloudflare dengan IP public 18.140.120.171 yang telah terintegrasi dengan Docker Container yang menggunakan port:80.



Gambar 2. Topologi Sistem

2.3 Rancangan pengujian

Setelah semua sistem telah dibangun maka selanjutnya Implementasi, pengujian dan analisis hasil pengujian dibahas di bab IV., adapun rancangan pengujiannya adalah;

1. Menyediakan domain beserta hostingan aktif tugasakhir95, domain ini berfungsi sebagai tamplate *website* yang akan diterapkan *CDN* yang sudah terintegrasi dengan *Docker Container*
2. Menkonfigurasi alamat domain dengan Cloudflare, konfigurasi ini berfungsi agar *cloudflare* dapat membaca semua konten dalam situs *website* tersebut.
3. Menginstal Docker Container pada sistem operasi Linux
4. Mengintegrasi antara *CDN* dengan Docker Container

3. HASIL DAN PEMBAHASAN

3.1. Tampilan Implementasi *Docker Container*

Pada tahap ini dapat dilihat pada gambar 3 menggunakan terminal yang terdapat pada linux , terminal sendiri berfungsi sebagai wadah untuk mengimplementasikan docker. Perintah apt-get update berfungsi untuk melakukan pembaruan beberapa komponen yang terdapat dalam terminal bertujuan agar instalasi docker berjalan dengan aman. Pembaruan komponen seperti gambar 3, menunjukan bahwa pembaruan yang diperlukan package pada terminal telah diupdate.

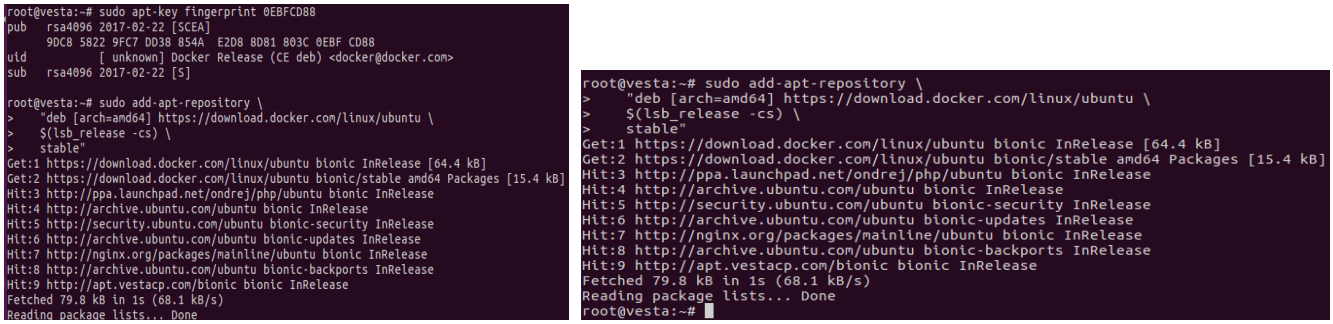
```
root@vesta:~# apt-get update
Hit:1 http://ppa.launchpad.net/ondrej/php/ubuntu bionic InRelease
Hit:2 http://nginx.org/packages/mainline/ubuntu bionic InRelease
Hit:3 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:4 http://archive.ubuntu.com/ubuntu bionic InRelease
Hit:5 http://archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:6 http://archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:7 http://apt.vestacp.com/bionic bionic InRelease
Reading package lists... Done
```

Gambar 3. Pembaruan Komponen Di Dalam Terminal Yang Diperlukan Docker

Tahap selanjutnya adalah menambahkan GPG key dari official repository docker ke dalam sistem dapat dilihat pada gambar 4. (a),(b) dan (c). tahap ini berfungsi untuk memperbaharui package docker yang versi lama ke versi terbaru.

```
root@vesta:~# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
OK
```

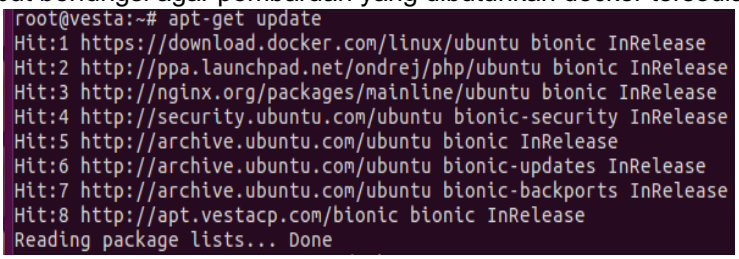
(a)



(b) (c)

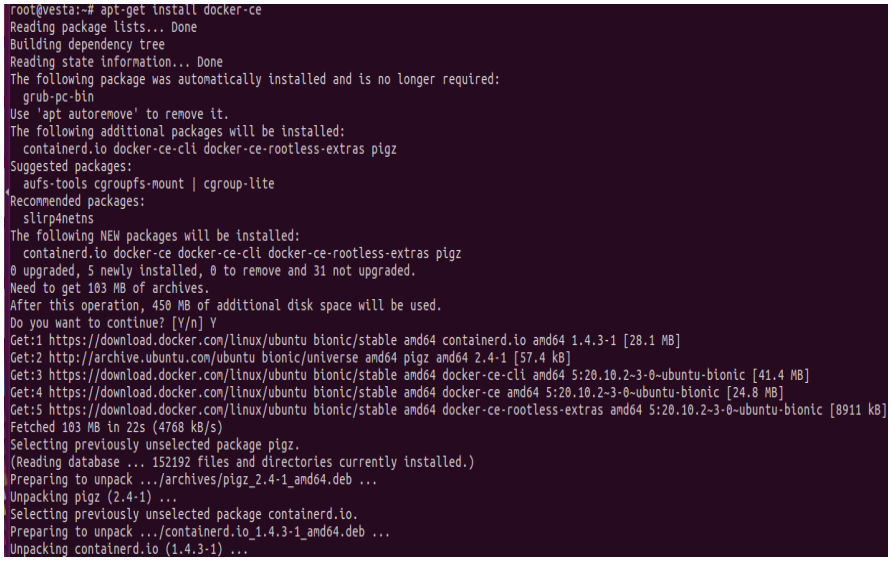
Gambar 4. Penambahan GPG Key Agar Docker Tersinstal Dengan Versi Yang Terbaru

Selanjutnya proses mengupdate kembali terminal agar docker berjalan dengan baik, dengan perintah apt-get-update Dilihat pada gambar 5. mengupdate kembali terminal setelah menambahkan GPG key ditahap sebelumnya, proses pengupdatean tersebut berfungsi agar pembaruan yang dibutuhkan docker tersedia pada linux.



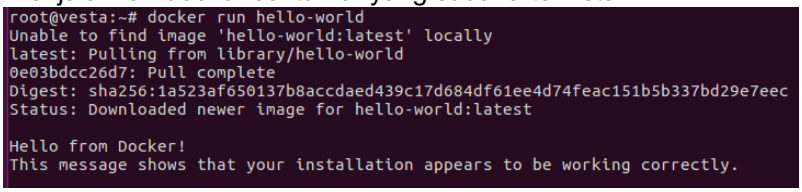
Gambar 5. Proses Mengupdate Terminal

Setelah proses penambahan GPG dan pengupdatean selesai, maka tahap selanjutnya yaitu menginstal Docker Container. Docker Container sendri berfungsi sebagau mengelola Container dan membuat sebuah image. dengan perintah apt-get apt install docker-ce, perintah tersebut untuk mendownload beberapa package yang dibutuhkan docker.



Gambar 6. Proses Penginstallan Docker

Proses selanjutnya menginput perintah *docker run hello-world*. Fungsi dari perintah docker run *docker run hello-world* untuk menguji atau menjalankan *docker container* yang sudaha terinstal.



Gambar 7. Proses Menjalankan Docker Container

Tahap selanjutnya menginput perintah *docker ps*, perintah ini berfungsi untuk melihat *Container* yang sedang berjalan. Berikut daftar *Container* dilihat pada gambar 8 di bawah ini.

```
root@vesta:~# docker ps
CONTAINER ID   IMAGE               COMMAND                  CREATED         STATUS         PORTS
 NAMES
049b00c13033   seqvence/static-site   "/bin/sh -c 'cd /usr…"   23 seconds ago   Up 22 seconds   0.0.0.0:49154->80/tcp, 0.0.0.0:49153->443/tcp
 frosty_nobel
```

Gambar 8. Container Yang Sudah Berjalan

Pada tahap selanjutnya dapat dilihat pada gambar 9.ifconfig berfungsi untuk melihat ip yang kita gunakan berfungsi untuk memanggil atau memastikan docker container sudah aktif.

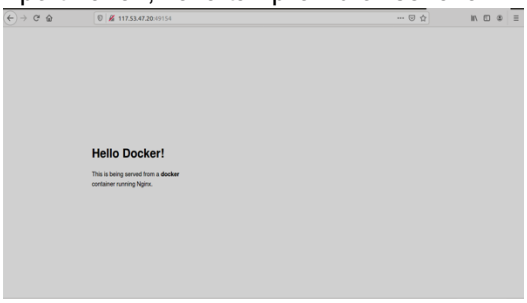
```
root@vesta:~# ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:c1ff:fed1:b450  prefixlen 64  scopeid 0x20<link>
        ether 02:42:c1:d1:b4:50  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5  bytes 446 (446.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 117.53.47.20  netmask 255.255.255.0  broadcast 117.53.47.255
        inet6 fe80::1469:39ff:fe0c:6de8  prefixlen 64  scopeid 0x20<link>
        ether 16:69:39:0c:6d:e8  txqueuelen 1000  (Ethernet)
        RX packets 506873763  bytes 32044596494 (32.0 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12146258  bytes 1608579986 (1.6 GB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
```

Gambar 9. Perintah ifconfig

Pada gambar 10 yang terlihat dibawah ini mengakses *Container* pada halaman browser, ketika memanggil ip yang digunakan *117.53.47.20* dengan port 49154,maka tampilan browser akan muncul seperti pada gambar 10.



Gambar 10. Docker Container Berhasil Dijalankan.

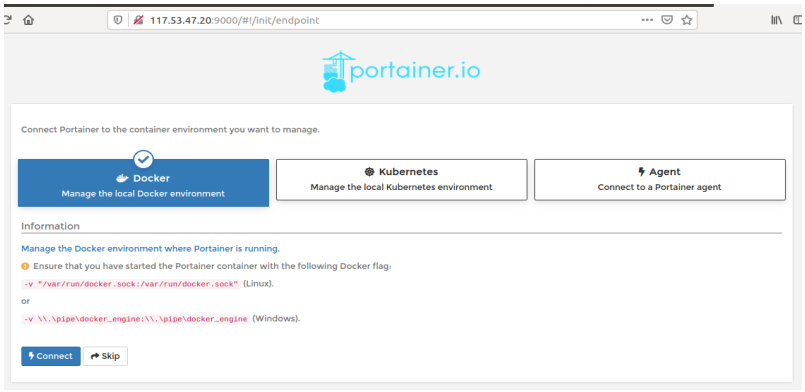
3.3. Tampilan Konfigurasi Portainer.io

Portainer adalah *Simple Management UI For Docker*, portainer berfungsi untuk mengelola Docker Image, kontainer, jaringan dan volume dari masing-masing kontainer melalui web dashboard yang sederhana. Pada tahap selanjutnya dapat dilihat pada gambar 11. yaitu menginstal Portainer dengan port *9000*. Port *9000* ini berfungsi untuk mengakses portainer pada browser.

```
root@vesta:~#  docker volume create portainer_data
portainer_data
root@vesta:~#  docker run -d -p 8000:8000 -p 9000:9000 --name=portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portai
ner_data:/data portainer/portainer-ce
Unable to find image 'portainer/portainer-ce:latest' locally
latest: Pulling from portainer/portainer-ce
b890dbc4eb27: Pull complete
81378af8dad0: Pull complete
Digest: sha256:21713e42233ee953b4cd4e6e8b1e4b6c43ebe2ca1c2dc762824a1866fdb91d3e
Status: Downloaded newer image for portainer/portainer-ce:latest
c85e48a3893e9b3258726be0fed1a338915743769c2046da70c0eb2f1f3a52cf
root@vesta:~#
```

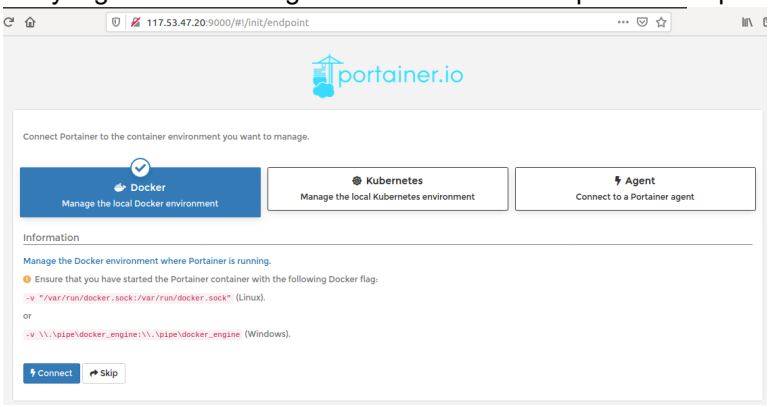
Gambar 11. Install Portainer

Setelah menginstal Portainer pada linux tahap selanjutnya yaitu mengakses portainer dengan menggunakan *IP 117.53.47.20* default dengan port *9000* yang sudah ditentukan pada tahap sebelumnya, portainer dapat diakses pada halaman browser, dapat dilihat pada gambar 12 seperti dibawah ini.



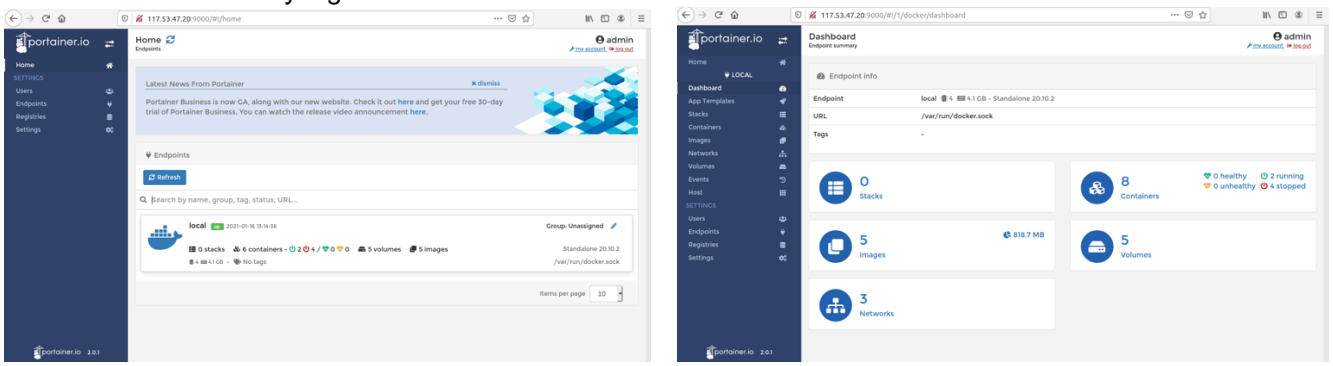
Gambar 12. Akses Portainer Pada Browser

Pada tahap selanjutnya dapat dilihat pada gambar 13.hubungkan portainer untuk memanajemen docker di local environment. Konfigurasi yang dilakukan ini agar Docker Container dapat dikelola pada Portainer



Gambar 13.Menghubungkan portainer dengan docker

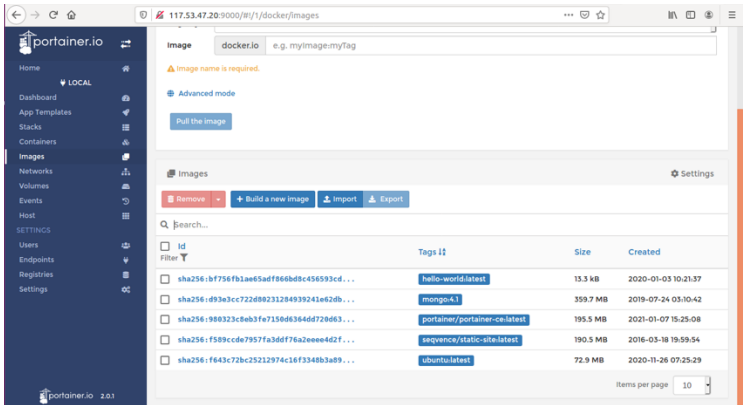
Pada proses selanjutnya menampilkan halaman awal dan halaman Dashboard. Pada halaman ini dapat memanajemen beberapa atau mengelola Docker Image, container, jaringan dan volume dari masing-masing container melalui web dashboard yang sederhana.



(a) (b)

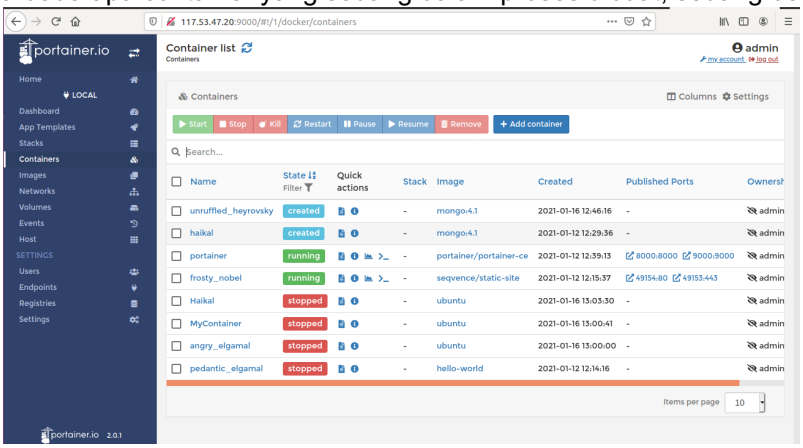
Gambar 14. Tampilan Halaman Awal Dan Halaman Dashboard

Pada tahap selanjutnya dapat dilihat pada gambar 15, halaman images pada halaman ini dapat membuild atau membuat suatu image serta dapat mengahapus,mengimport dan export teresebut.



Gambar 15. Halaman Manajemen Images

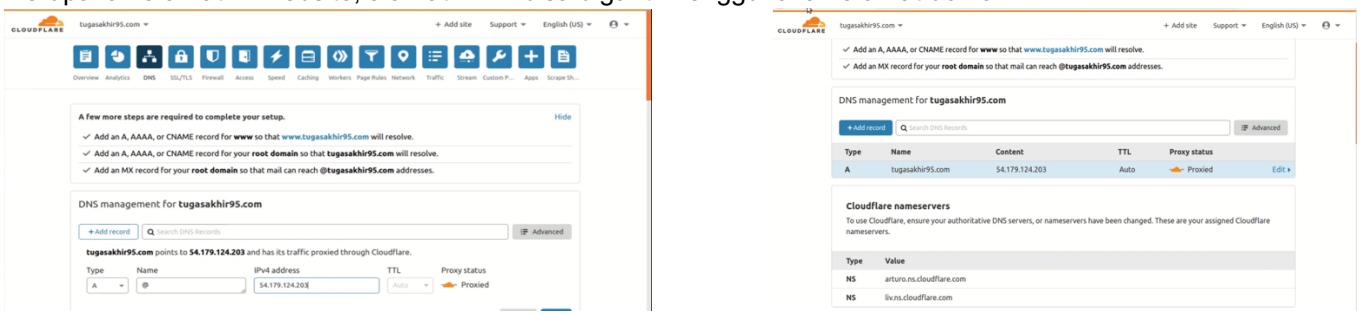
Pada proses selanjutnya dapat dilihat container list, pada halaman ini dapat memanajemen container seperti menambah container, menghapus container, menjalankan container. Terdapat beberapa container yang telah dibuat, pada gambar tersebut ada beberapa container yang sedang dalam proses dibuat, sedang berjalan, dan sedang stop.



Gambar 16 Halaman Container List

3.3. Tampilan Konfigurasi CDN(*Content Delivery Network*)

Proses selanjutnya dapat dilihat pada gambar 17. setting DNS, type yang digunakan type A, name yang digunakan adalah @ berfungsi untuk menggunakan nama domain tugasaakhir95, IPv4 addrees *54.179.124.203* merupakan alamat IP website, alamat IP ini bisa diganti menggunakan alamat domain.

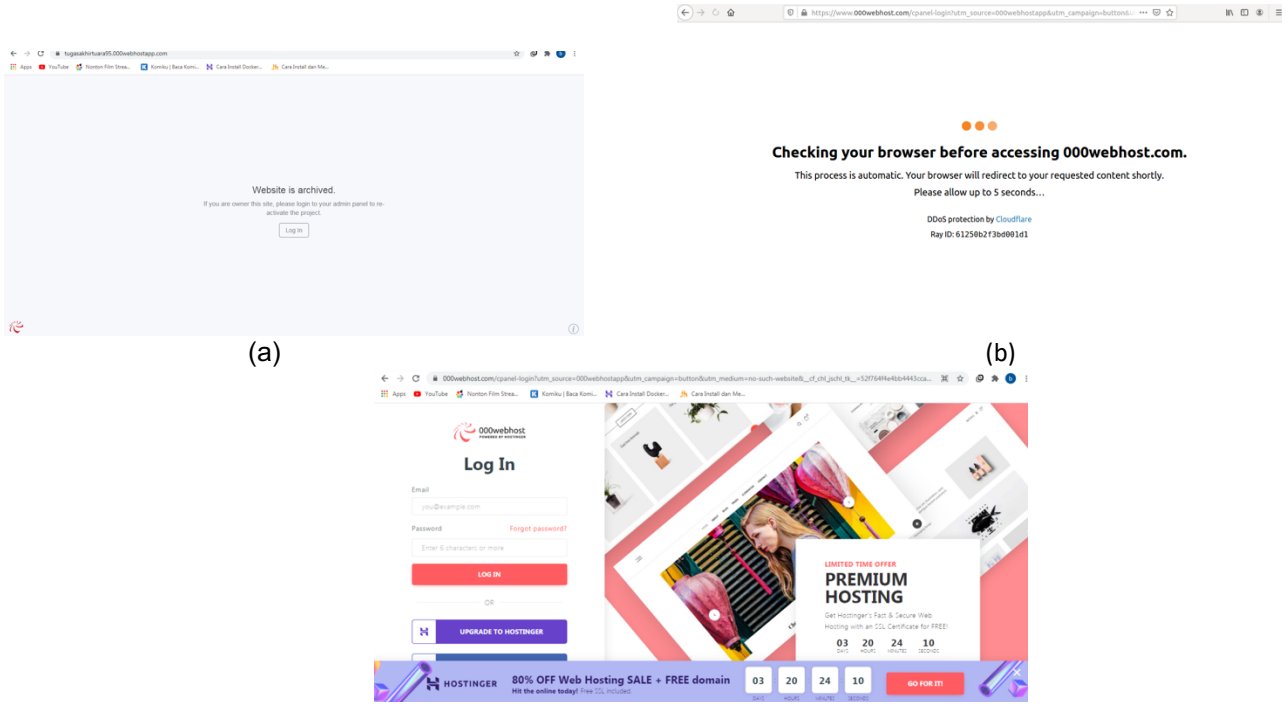


(a) (b)

Gambar 17. (A) Setting DNS (B) Hasil Setting DNS

3.5. Tampilan Tahapan Pengujian

Setelah proses penginstalan Docker dan mengkonfigurasi CDN, maka selanjutnya dilakukan pengujian terhadap system, proses ini bertujuan untuk mengetahui apakah CDN dan Docker Container sudah terintegrasi dengan baik, adapun hasil pengujian implementasi CDN yang sudah terintegrasi dengan Docker Container pada halaman *website* dapat dilihat pada gambar 18, (a),(b) dan (c) dibawah ini.

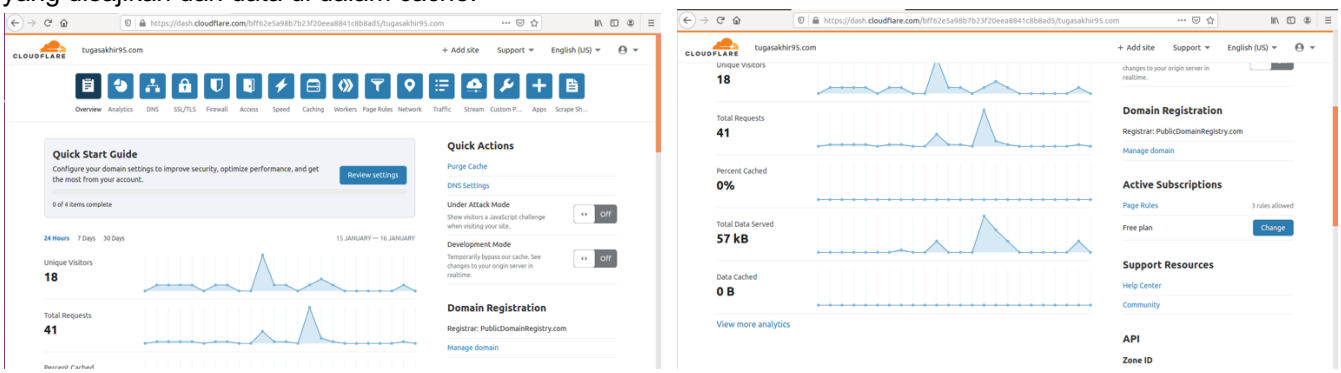


(a) (b)

(c)

Gambar. 18, (a),(b),dan (c) Tampilan Website CDN dan Dcoker Container

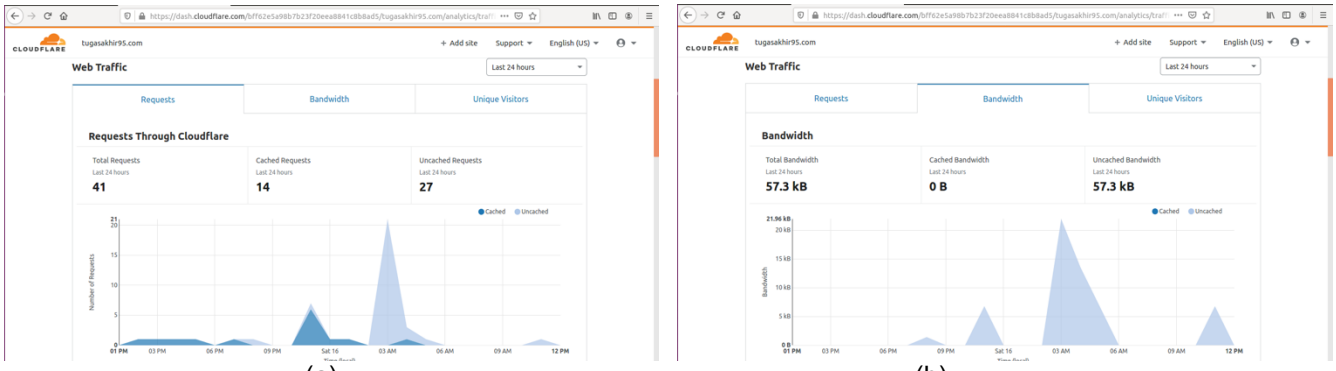
Pada gambar 19 (a) dan (b), adalah halaman overview halaman overview ini dapat mengetahui berapa persentasi dari pengunjung unik, total permintaan, berapa persen data yang disimpan sementara (cache),total data yang disajikan dan data di dalam cache.



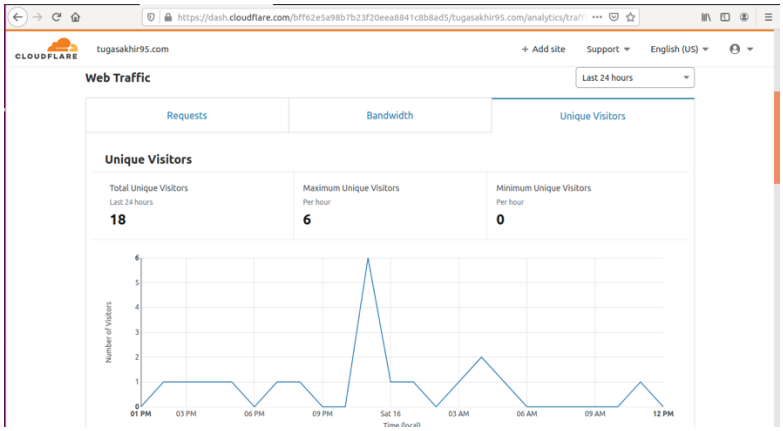
(a) (b)

Gambar 19. (A) Dan(B) Halaman Overview

Selain beberapa pengujian diatas, selanjutnya mengecek *web traffic*, pada halaman ini terdapat presentasi *request*,*bandwitch*, dan *unique request*, dimana pada halaman *request* berfungsi untuk melihat permintaan akses, halaman *bandwidth* berfungsi untuk mengecek berapa presentasi *bandwidth* atau jumlah konsumsi transfer data yang dihitung dalam satuan waktu *bit per second* (bps), dan halaman *unique visitor* berfungsi untuk mengecek berapa presentasi pengunjung unik. Dapat dilihat pada gambar 20. (a),(b) dan (c). Pada gambar 20 terdiri dari beberapa user, dimana beberapa user tersebut mengakses internet yang berdomain tugasakhir95.com menggunakan cloudflare dengan IP public 18.140.120.171 yang sudah terintegrasi dengan Docker Container yang menggunakan port:80



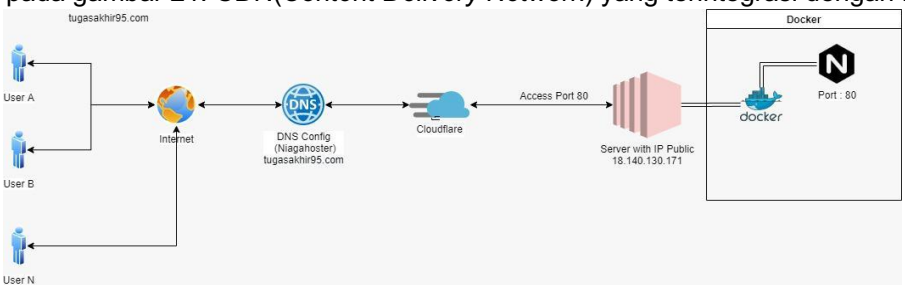
(a) (b)



(c)

Gambar 20. (A) Halaman Presentasi Request,(B) Halaman Presentasi Bandwidth, (C) Halaman Presentasi Unique Visitor

Dapat dilihat pada gambar 21. CDN(*Content Delivery Network*) yang terintegrasi dengan Docker Container



Gambar 21. Alur Mengakses Tugasakhir95

4. KESIMPULAN

Dari hasil implementasi dan pengujian secara keseluruhan dapat disimpulkan,Penulis berhasil membangun sebuah system CDN(*Content Delivery* Network) dapat bekerja dengan baik sebagaimana fungsinya dan telah terintegrasi dengan docker container dan Penulis juga berhasil menguji dan menganalisa bagaimana cara mengimplementasikan CDN (*Content Delivery Network*) menggunakan Cloudflare yang terintegrasi dengan Docker Countainer.

Daftar Pustaka

[1] Reska Setiawan, 2009. Penggunaan Internet sebagai Teknologi Informasi di kalangan Mahasiswa Ekonomi Akuntansi Universitas Muhammadiyah Surakarta. Skripsi. Fakultas Ekonomi Akuntansi. Universitas Muhammadiyah Malang

[2] I Gede Putu Krisna Juliharta.”Distribusi Konten Web Server Menggunakan Motode Content Delivery Network”. Jurnal Sistem Dan Informatika, Vol.10, No. 1, November 2015

[3] M. Fadlulloh Romadlon Bik. “Implementasi *Docker* untuk pengelolaan banyak aplikasi *web*”. Jurnal Manajemen Informatika, Vol.7, No. 2, Tahun 2017

[4] Dewi Laksmiati. “Implementasi *Content Delivery Network (CDN)* untuk optimasi kecepatan akses *website*”. Jurnal Akrab Juara, Vol 5, No. 1, Februari 2020

[5] Sahat Parulian Sitorus, 2017. Analisis Kinerja *Content Delivery Network*. Tesis. Fakultas Ilmu Komputer dan Teknologi Informasi. Universitas Sumatera Utara, Medan.

[6] Dewi Laksmiati. “Implementasi *Content Delivery Network (CDN)* untuk optimasi kecepatan akses *website*”. Jurnal Akrab Juara, Vol 5, No. 1, Februari 2020

[7] M. Ray Akbar Mutalibov.”Pengaruh content delivery network (CDN) terhadap potensi cyberwar di Indonesia”, Bandung, 2013

[8] Dewi Estri Jayanti , Rusydi Umar, Imam Riadi.”Implementasi *Cloudflare Hosting* untuk Kecepatan Akses Pada *Website Trading*, Jurnal Sisfotenika, Vol.10, No. 2, Juli 2020

[9] Addri Pershance And Taga, 2017. Rancang Bangun *Web Hosting* menggunakan *Docker Container* dan *Clustering* pada *Coreos*: *Docker container*. Tugas Akhir. Fakultas Teknik. Universitas Muhammadiyah Malang.

[10] Endah Sri Maulana Sardi, 2017.Implementasi Teknik Virtualisasi *Container* dengan *Docker* untuk pengelolaan aplikasi *web* di Dinas Komunikasi dan Informatika Kota Payakumbuh. Tugas Akhir. Program Studi Teknik Komputer. Jurusan Teknologi Informasi. Politeknik Negeri Padang.

[11] Saleh Dwiyatno, Edy Rakhmat, Oki Gustiawan. “Implementasi Virtualisasi *Server* Berbasis *Docker Container*”. Jurnal Posisko, Vol.7, No 2, September 2020