SKRIPSI
IMPLEMENTASI LAMP STACK DAN EMAIL SERVER MENGGUNAKAN DOCKER DI PT.PUNDI MAS BERJAYA
(MAGANG BERSERTIFIKAT KAMPUS MERDEKA)

Aldzikri Dwijayanto Prathama
NIM: 195410189

PROGRAM STUDI INFORMATIKA
PROGRAM SARJANA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA
YOGYAKARTA
2023

SKRIPSI

IMPLEMENTASI LAMP STACK DAN EMAIL SERVER MENGGUNAKAN DOCKER DI PT.PUNDI MAS BERJAYA

(MAGANG BERSERTIFIKAT KAMPUS MERDEKA)

Diajukan sebagai salah satu syarat untuk menyelesaikan studi

Program Sarjana

Program Studi Informatika

Fakultas Teknologi Informasi

Universitas Teknologi Digital Indonesia

Yogyakarta

Disusun Oleh

Aldzikri Dwijayanto Prathama

NIM: 195410189

PROGRAM STUDI INFORMATIKA

PROGRAM SARJANA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA

YOGYAKARTA

2023

HALAMAN PERSETUJUAN UJIAN SKRIPSI

Judul	:	Implementasi Lamp Stack dan Email Server Menggunakan Docker di PT.Pundi Mas Berjaya (Magang Bersertifikat Kampus Merdeka)
Nama	:	Aldzikri Dwijayanto Prathama
NIM	:	195410189
Program Studi	:	Informatika
Program	:	Sarjana
Semester	:	Gasal
Tahun akademik	:	2022/2023

Telah diperiksa dan disetujui untuk diujikan di hadapan Dewan Penguji Skripsi.

Yogyakarta, 2023

Dosen Pembimbing,

Danny Kriestanto, S.Kom., M.Eng.

NIDN: 0503068002

HALAMAN PENGESAHAN

SKRIPSI

IMPLEMENTASI LAMP STACK DAN EMAIL SERVER MENGGUNAKAN DOCKER DI PT.PUNDI MAS BERJAYA

(MAGANG BERSERTIFIKAT KAMPUS MERDEKA)

Telah dipertahankan di depan Dewan Penguji Skripsi dan dinyatakan diterima untuk memenuhi sebagian persyaratan guna memperoleh Gelar Sarjana Komputer

Program Studi Informatika

Fakultas Teknologi Informasi

Universitas Teknologi Digital Indonesia

Yogyakarta

Yogyakarta, ... 2023

Dewan Penguji	NIDN	Tanda tangan
1. Ir. Muhammad Guntara, M.T.	0509066101	.....
2. Danny Kriestanto, S.Kom., M.Eng.	0503068002	.....
3. Adi Kusjani, S.T, M.Eng.	0515067501	.....

Mengetahui

Ketua Program Studi Informatika

Dini Fakta Sari, S.T., M.T.

NIDN: 0507108401

PERNYATAAN KEASLIAN SKRIPSI

Dengan ini saya menyatakan bahwa naskah skripsi ini belum pernah diajukan untuk memperoleh gelar Sarjana Komputer di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara sah diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 23 Februari 2023

Aldzikri Dwijayanto Prathama
NIM: 195410189

HALAMAN PERSEMBAHAN

Bismillahirrahmanirrahim, Alhamdulillahi robbil'alamiin. Puji Syukur saya panjatkan kepada Allah SWT, yang telah memberikan kesehatan, rahmat dan hidayah, sehingga saya masih diberikan kesempatan untuk menyelesaikan skripsi ini sebagai salah satu syarat untuk mendapatkan gelar kesarjanaan. Skripsi ini saya persembahkan untuk :

1. Allah SWT yang selalu melimpahkan rahmat dan hidayah-Nya.
2. Keluarga saya, kedua orang tua saya, dan kakak saya yang telah memberikan do'a dan dukungan hingga saat ini.
3. Teman-teman Dimas, Fadhil, Rizqul, Putra, dan teman-teman lainnya yang telah menemani dan membantu menempuh selama jenjang pendidikan ini.
4. Semua pihak yang tidak dapat saya sebutkan satu persatu yang telah membantu terselesaikannya skripsi ini.

HALAMAN MOTO

”Mencari apa-apa untuk hidup itu boleh saja
tapi jangan lupa hidup itu untuk apa”

KATA PENGANTAR

Puji syukur saya panjatkan kepada Allah Subhanahu Wa Ta'ala yang telah memberikan nikmat dan karunia sehingga saya dapat menyelesaikan skripsi dengan judul "Magang Bersertifikat Kampus Merdeka Implementasi LAMP Stack dan Email Server Menggunakan Docker di PT.Pundi Mas Berjaya".

Skripsi ini ditujukan sebagai salah satu syarat untuk menyelesaikan studi program sarjana, Program Studi Informatika. Dalam pengerjaan laporan ini telah melibatkan banyak pihak, oleh sebab itu, saya sampaikan rasa terima kasih sedalam-dalamnya kepada:

- Bapak Ir. Totok Suprawoto, M.M., M.T. Sebagai Rektor Universitas Teknologi Digital Indonesia.
- Ibu Dini Fakta Sari, S.T., M.T dan Ibu Femi Dwi Astuti, S.Kom., M.Cs. Selaku Ketua dan Sekretaris Program Studi Informatika.
- Danny Kriestanto, S.Kom., M.Eng. selaku pembimbing yang telah meluangkan waktunya untuk membimbing saya.
- Teman-teman mahasiswa Universitas Teknologi Digital Indonesia yang tidak bisa penulis sebutkan satu persatu yang telah memberikan dukungan dan semangat untuk menyelesaikan skripsi ini.

Yogyakarta, ... 2023

Aldzikri Dwijayanto Prathama
NIM: 195410189

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

INTISARI

PT.Pundi Mas Berjaya adalah salah satu penyedia solusi perangkat lunak di pasar global. PT. Pundi Mas Berjaya memiliki pelanggan yang tersebar di beberapa negara, dan pelanggan dari perusahaan yang bergerak di masing-masing bidang yang berbeda. Dengan banyaknya pelanggan dengan asal negara dan bidang yang bermacam-macam, tentunya kebutuhan aplikasi perusahaan tersebut berbeda-beda pula. Oleh karena itu PT.Pundi Mas Berjaya membutuhkan solusi untuk mengisolasi aplikasi dan layanan yang dikembangkan dan di-*host*, serta dapat di-*deploy* dengan cepat dan mudah.

Pada penelitian ini adalah membahas cara membangun sebuah server LAMP dan email, dengan metode *containerization* menggunakan Docker. Sehingga memudahkan manajemen, dan *deployment* aplikasi.

Penelitian ini menghasilkan kesimpulan bahwa dengan menggunakan Docker untuk manajemen dan *deployment* aplikasi menjadi lebih mudah dan tidak memakan waktu yang lama dibandingkan dengan metode manual.

Kata Kunci: *Container, Docker, LAMP, Email server*

ABSTRACT

PT.Pundi Mas Berjaya is one of the leading software solution providers in the global market. PT. Pundi Mas Berjaya has customers spread across several countries, and customers from companies engaged in each of the different fields. With so many customers from various countries and fields, surely they have applications requirements that varies too. Therefore PT.Pundi Mas Berjaya needs a solution to isolate the application and their services, and can be deployed quickly and easily.

In this research I tried to build a LAMP and email server, with the containerization method using Docker. So that makes applications management and deployment is much more easier.

This study resulted in the conclusion that by using Docker applications management and deployment become easier and faster compared to manual method

Keywords: *Container, Docker, LAMP, Email server*

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Perkembangan koneksi internet yang semakin cepat dan terjangkau mengubah cara bagaimana dunia bekerja. Dari bidang komunikasi dan hiburan, hingga bidang komersial dan pendidikan, internet telah menjadi hal yang umum di masyarakat modern. Di ranah bisnis, internet menciptakan peluang baru bagi perusahaan untuk menjangkau pasar yang lebih luas dan beroperasi dengan lebih efisien dengan cara yang inovatif(Parinda et al. 2023). Oleh karena itu perusahaan perlu merangkul dan mengintegrasikan internet ke dalam infrastruktur yang dimiliki sebagai sarana agar tetap unggul dalam ranah bisnis yang terus berkembang.

PT.Pundi Mas Berjaya adalah salah satu penyedia solusi perangkat lunak di pasar global yang memberikan solusi bisnis mengadopsi teknologi informasi terkini. Solusi berbasis layanan untuk para pelanggan yang tersebar di beberapa negara dan memiliki model pengembangan on-site dan off-site. Perusahaan ini merancang dan mengembangkan banyak solusi di bidang properti, otomotif, transportasi, pengirimanan makanan, pengiriman barang dan ecommerce.

Tentunya dengan memiliki klien yang banyak dan berasal dari perusahaan yang beragam PT.Pundi Mas Berjaya memiliki masalah tersendiri. Setiap perusahaan memiliki *requirements* dan *tools* masing-masing yang unik untuk memenuhi kebutuhan infrastruktur IT perusahaan. Dengan begitu tiap-tiap perangkat lunak yang dibuat dan di-*host* oleh PT.Pundi Mas Berjaya membutuhkan *requirements* yang berbeda pula. Misalnya, perangkat lunak untuk perusahaan A menggunakan LAMP *stack* dengan PHP versi 7, sedangkan perangkat lunak untuk perusahaan B menggunakan LAMP *stack* dengan PHP versi 8. Selain itu tiap layanan yang di-*host*

harus dengan mudah direplikasi untuk membuat *instance* baru, contohnya membuat *instance* email server baru untuk pelanggan baru. Sehingga PT.Pundi Mas Berjaya perlu mengisolasi tiap-tiap perangkat lunaknya sehingga dapat bekerja dengan baik dan menghindari *conflict* antar versi serta dapat disiapkan dan dijalankan dengan cepat.

Maka dari itu, PT.Pundi Mas Berjaya mengimplementasikan *containerization* menggunakan docker dengan tujuan untuk memaketkan dan mengisolasi perangkat lunaknya, mengotomatisasikan *deployment* dan manajemen aplikasi, serta memastikan perangkat lunak dapat dijalankan di komputer yang berbeda.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang diuraikan di atas, maka dapat dirumuskan pokok permasalahannya, yaitu "Bagaimana cara membangun LAMP-*stack* dan email server yang terisolasi yang dapat dengan cepat dan mudah direplikasi dengan menggunakan Docker".

1.3 Ruang Lingkup

Berikut ruang lingkup untuk dapat mencapai sasaran dan tujuan yang diharapkan dari peneliti :

1. Email server yang dibangun bisa mengirim email ke akun email Gmail tanpa ditandai sebagai spam.
2. Sistem dibangun memanfaatkan docker.
3. Menggunakan Image LAMP dan email server yang sudah tersedia di Docker Hub.

4. Sistem operasi yang digunakan untuk menjalankan docker adalah Ubuntu server.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah menghasilkan server LAMP yang berfungsi dan server email yang dapat mengirim email ke akun email Gmail tanpa ditandai sebagai spam menggunakan docker.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah menghasilkan konfigurasi docker compose yang mudah dilakukan *backup*, dan mudah direplikasi sehingga memudahkan pembangunan LAMP server dan email server.

1.6 Sistematika Penulisan

Berikut merupakan sistematika penilisan skripsi yang akan dibuat:

1. **Bab I Pendahuluan**

   Pada bab ini berisi tentang penjelasan mengenai latar belakang masalah, rumusan masalah, ruang lingkup, tujuan penelitian, manfaat penelitian, dan sistematika penulisan

2. **Bab II Tinjauan Pustaka dan Dasar Teori**

   Pada bab ini berisi tentang pembahasan sumber pustaka yang digunakan sebagai pedoman perancangan penelitian yang berhubungan dengan penelitian yang sedang diteliti.

3. **Bab III Metode Penelitian**

   Pada bab ini berisi tentang pembahasan analisis kebutuhan, bahan/data, dan juga perancangan sistem yang akan digunakan.

4. **Bab IV Implementasi dan Pembahasan**

   Pada bab ini menguraikan tentang pembuatan aplikasi yang merupakan implementasi dari hasil analisa dan perancangan, pengujian sistem, dan juga kesimpulan.

5. **Bab V Penutup**

   Pada bab ini berisi kesimpulan dari hail pembahasan penerapan sistem dan saran-saran guna pengembang sistem yang telah dibuat.

BAB 2

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Dalam mengimplementasikan LAMP *stack* dan email server dengan menggunakan Docker, sebagai pedoman dan pembanding maka digunakan beberapa pustaka yang berkaitan dengan *Docker*. Pustaka yang digunakan sebagai rujukan antara lain:

Tabel 2.1: Tabel Pustaka

Penulis	**Topik**	**Hasil Penelitian**	**Teknologi yang digunakan**
Wijayanto (2018)	'Tehnologi Platfrom Virtualisasi Untuk Aplikasi Lowongan Kerja'	Menghasilkan aplikasi lowongan kerja yang dibuat menjadi docker images yang dapat berjalan di container docker sehingga dapat dimanfaatkan di penyedia *platform as a service* (PAAS).	Docker
Dwiyatno (2020)	'Implementasi virtualisasi server berbasis docker container'	Mengimplementasikan sistem virtualisasi server berbasis docker container di SMK Negeri 1 Rangkasbitung dan pengujian penggunaan sumber daya dari penggunaan Docker sebagai sistemt virtualisasi	Docker, Nginx, Apache Web Server, Apache Jmeter
Megantara et al. (2021)	'IMPLEMENTASI DOCKER DALAM MEMBANTU VIRTUAL LAB PADA UNIVERSITAS'	Menghasilkan sistem untuk membantu mahasiswa melakukan praktek pemrograman pada saat pembelajaran jarak jauh. Dengan sistem ini mahasiswa dapat mengerjakan tugas di beragam perangkat dan memudahkan dosen memeriksa hasil praktek.	Docker, Jupyter Notebook
Prasetyo dan Wijaya (2021)	'Analisa dan Implementasi Microservice pada Container Menggunakan Docker'	Analisa performa microservice yang diimplementasikan di docker container. Dalam penelitian ini disimpulkan docker memiliki efisiensi yang lebih baik dibanding virtualisasi hypervisor.	Docker, Apache Jmeter

Penulis	Topik	Hasil Penelitian	Teknologi yang digunakan
Furnama (2022)	'Implementasi Arsitektur Microservices Pada Sistem Backend Pembayaran Terintegrasi Menggunakan Docker'	Menghasilkan sistem *backend* pembayaran terintegrasi Midtrans dengan arsitektur microservice yang berjalan di atas docker.	Docker, Midtrans
Saputra dan Rismayadi (2021)	'IMPLEMENTASI CLOUD STORAGE MENGGUNAKAN OWNCLUOD DAN DOCKER'	Penelitian ini menghasilkan sistem cloud storage menggunakan OwnCloud dan Docker. Dengan menggunakan Docker, pemasangan OwnCloud tidak mempengaruhi sistem *host*, selain itu Docker juga menghemat penggunaan resource.	Docker, OwnCloud
Kristiawan, Wahanani dan Idhom (2022)	'Implementasi Reverse Proxy Pada Hosting Web Server di Docker Container'	Pada penelitian ini menghasilkan sistem web hosting dengan menggunakan Wordpress sebagai CMS yang memanfaatkan nginx sebagai web servernya, sedangkan untuk reverse proxy digunakan Apache Web Server.	Docker, Wordpress, Nginx
Aldzikri Dwijayanto Prathama (Diusulkan)	Implementasi Lamp Stack Dan Email Server Menggunakan Docker di PT.Pundi Mas Berjaya (Magang Bersertifikat Kampus Merdeka)	Pada penelitian ini adalah membahas cara membangun sebuah server LAMP dan email, dengan metode *containerization* menggunakan Docker. Sehingga memudahkan manajemen, dan *deployment* aplikasi.	Docker, LAMP-stack, Postfix, Dovecot

2.2 Dasar Teori

2.2.1 PT.Pundi Mas Berjaya

PT.Pundi Mas Berjaya adalah salah satu penyedia solusi perangkat lunak di pasar global yang memberikan solusi bisnis mengadopsi teknologi informasi terkini. Solusi berbasis layanan untuk para pelanggan yang tersebar di beberapa negara dan memiliki model pengembangan on-site dan off-site. Sejak berdiri pada tahun 2014, Perusahaan telah merancang, mengembangkan dan digunakan banyak solusi di bidang properti, otomotif, transportasi, pengirimanan makanan, pengiriman barang dan ecommerce.

PT.Pundi Mas Berjaya memiliki tim yang kuat dengan ukuran pelaksanaan proyek besar dan mengerahkan seluruh kemampuannya untuk memberikan pelayanan yang berkualitas tinggi kepada pelanggan. Salah satu kunci keberhasilan PT.Pundi Mas Berjaya adalah adaptasi perusahaan tersebut terhadap berbagai macam dan aneka kebutuhan akan sistem informasi komputer sehingga menghasilkan Manajemen Kualitas, Manajemen Proyek, kebutuhan Infrastruktur, dan lainnya dengan sempurna untuk memberikan kepuasan kepada para pelanggan PT.Pundi Mas Berjaya. Pelayanan PT.Pundi Mas Berjaya sifatnya terpadu menggunakan media responsif website, mobile website, mobile apps untuk mendukung aktifitas yang lebih optimal.

PT.Pundi Mas Berjaya memiliki tim berpengalaman di bidang:

1. Manajemen teknis website
2. Manajemen teknis mobile application
3. Manajemen bisnis untuk aplikasi yang sedang dibangun
4. Manajemen kendali mutu untuk aplikasi yang dijalankan

Sedangkan fokus bisnis dari perusahaan ini antara lain:

1. Aplikasi properti online kolaborasi dalam dan luar negeri
2. Aplikasi transportasi berbasis aplikasi
3. Aplikasi bank data untuk pebisnis
4. Aplikasi pelayanan pengantaran makanan
5. Aplikasi pelayanan pengiriman bunga/parcel
6. Aplikasi Deal untuk aneka voucher
7. Aplikasi e-commerce

2.2.2 Container

Menurut Shah dan Dubaria (2019) *Containerization* adalah cara untuk menjalankan beberapa perangkat lunak yang berbeda di satu mesin. Yang masing-masingnya berjalan di dalam lingkungan yang terisolasi yang disebut sebagai *container*. *Container* adalah lingkungan yang terisosali untuk perangkat lunak. Container mengikat semua berkas dan *library* yang dibutuhkan oleh aplikasi untuk berfungsi dengan baik.

2.2.3 Docker

Docker adalah sebuah project *open-source* yang menyediakan platform terbuka untuk developer maupun sysadmin untuk dapat membangun, mengemas, dan menjalankan aplikasi dimanapun di dalam sebuah container(Docker 2023).

Docker menggunakan arsitektur client-server. Dimana client dan docker berkomunikasi dengan daemon docker, yang melakukan suatu tindakan untuk membangun, menjalankan, dan mendistribusikan container docker. Client docker dan daemon dapat berjalan pada sistem yang sama. Client docker dan daemon berkomunikasi menggunakan REST API, melalui soket UNIX atau antarmuka jaringan(Docker 2023).

2.2.4 Docker Compose

Docker Compose adalah alat untuk mendefinisikan dan menjalankan aplikasi Docker multi-kontainer. Dengan Docker Compose, file YAML bisa digunakan untuk mengonfigurasi layanan aplikasi. Kemudian, hanya dengan satu perintah, Docker akan membuat dan memulai semua layanan dari konfigurasi YAML tersebut.

2.2.5 LAMP *stack*

LAMP *stack* adalah gabungan dari empat teknologi perangkat lunak berbeda yang digunakan oleh developer untuk membangun situs web dan aplikasi web. LAMP adalah singkatan dari sistem operasi Linux; server web Apache; server basis data

MySQL; dan bahasa pemrograman PHP. Keempat teknologi ini bersifat *open-source*, yang berarti LAMP dikelola komunitas dan tersedia secara bebas untuk digunakan oleh siapa saja. Developer menggunakan LAMP *stack* untuk membuat, meng-host, dan memelihara konten web. LAMP *stack* adalah solusi populer yang mendukung banyak situs web yang biasa gunakan saat ini(Amazon 2023).

2.2.6 Email Server

Email server adalah perangkat lunak yang berfungsi untuk mengirim dan menerima email. Email server biasanya menjadi seebutan untuk *Mail Transfer Agent* (MTA) dan *Mail Delivery Agent* (MDA), yang keduanya memiliki fungsi yang berbeda(Cloudflare 2023).

Pesan email dikirim dan diterima dengan menggunakan dua jenis email server: *outgoing mail server* atau *Mail Transfer Agent*, dan *incoming mail server* atau *Mail Delivery Agent*. MTA akan menerima pesan yang keluar dari *email client* pengirim, kemudian mengirimkannya ke MDA, yang mana bertanggung jawab untuk menyimpan sementara dan menyampaikan pesan ke *email client* penerima(Cloudflare 2023).

2.2.7 DNS TXT Record

TXT Record, Merupakan sebuah record DNS yang digunakan untuk menyimpan informasi berupa text pada domain. Teks pada TXT Records harus diformat sesuai dengan teknologi yang akan gunakan(Rosenbaum 1993).

2.2.8 DKIM

Merujuk pada dokumen RFC 6376, DKIM *(DomainKeys Identified Mail)* adalah teknik untuk memverifikasi keaslian pesan email dengan memeriksa tanda tangan digital yang disematkan di header pesan. Tanda tangan ini dihasilkan dengan

mengenkripsi hash konten pesan dengan *private key*, yang dapat diverifikasi oleh penerima menggunakan *public key* yang diterbitkan dalam data DNS TXT domain pengirim(Kucherawy, Crocker dan Hansen 2011).

Dengan DKIM email server penerima dapat memverifikasi bahwa pesan yang masuk tidak diubah saat transit dan berasal dari domain yang digunakan. DKIM *signature* dapat dibuat dengan menambahkan header ke pesan yang berisi informasi tentang pesan dan tanda tangan kriptografi dari isi pesan dan beberapa header. Server email penerima kemudian dapat menggunakan informasi di header DKIM dan kunci publik yang dipublikasikan di DNS domain pengirim untuk memverifikasi keaslian dan integritas pesan(Kucherawy, Crocker dan Hansen 2011).

2.2.9 SPF

Menurut dokumen RFC 7208, SPF *(Sender Policy Framework)* adalah protokol autentikasi email yang dirancang untuk mendeteksi dan mencegah spoofing email dengan cara memverifikasi identitas domain yang diklaim berasal dari email. SPF bekerja dengan memberikan daftar alamat IP atau nama host pengiriman resmi untuk domain di DNS, dan server email penerima dapat menggunakan informasi tersebut untuk memvalidasi keaslian pesan masuk(Kitterman 2014).

Saat email diterima, server email penerima memeriksa data SPF untuk domain pada header ”From” pesan untuk menentukan apakah pengirim diizinkan untuk mengirim email atas nama domain yang digunakan. Jika server pengirim tidak tercantum dalam catatan SPF, pesan tersebut mungkin akan ditandai sebagai spam atau bahkan langsung ditolak(Kitterman 2014).

2.2.10 DMARC

Menurut dokumen RFC 7489, DMARC *(Domain-based Message Authentication, Reporting & Conformance)* merupakan metode autentikasi, pengaturan dan pelaporan yang menggunakan kombinasi metode autentikasi SPF (Sender Policy Framework) dan DKIM (DomainKeys Identified Mail) untuk menentukan apakah pesan email itu sah. Dengan menggunakan DMARC, email server penerima dapat mengambil tindakan yang sesuai, seperti menolak atau mengkarantina pesan, berdasarkan peraturan yang dipublikasikan pemilik domain di DMARC-nya(Kucherawy dan Zwicky 2015).

DMARC juga memiliki mekanisme pelaporan bagi pemilik domain untuk menerima umpan balik tentang pesan yang lulus atau gagal dalam evaluasi DMARC, dan membantu pemilik domain memantau pengiriman email dan mendeteksi potensi terjadinya spoofing email(Kucherawy dan Zwicky 2015).

BAB 3

METODE PENELITIAN

3.1 Bahan/Data

Bahan dan data yang diperlukan pada penelitian ini adalah aplikasi web, dan email.

3.2 Analisis Kebutuhan Perangkat

Dalam proses pembuatan sistem ini, membutuhkan perangkat lunak dan perangkat keras yang digunakan sebagai *server* dan *client* untuk me-*remote server*, analisis kebutuhan dalam pembuatan sistem ini antara lain:

3.2.1 Kebutuhan Perangkat Lunak

1. **Perangkat Lunak Server**

   (a) Sistem Operasi Ubuntu Server

   (b) OpenSSH

   (c) Docker

   (d) Rsync

   (e) Vim

2. **Perangkat Lunak Client**

   (a) Sistem Operasi NixOS

   (b) OpenSSH

   (c) Rsync

   (d) Neovim

3.2.2 Kebutuhan Perangkat Keras

1. **Perangkat Keras Server**

   (a) Dell PowerEdge Rack Server

   (b) Switch

   (c) Router

2. **Perangkat Keras Client**

   (a) Laptop Lenovo ThinkPad T440p

   (b) CPU i7 4702MQ

   (c) RAM 8GB

   (d) SSD 250GB

3.2.3 Kebutuhan Lainnya

1. Domain Name

3.3 Kebutuhan Input

Kebutuhan masukan (input) merupakan sekumpulan data yang akan diproses oleh sistem. Adapun kebutuhan input yang dibutuhkan sistem ini yaitu:

1. Aplikasi web

2. Email

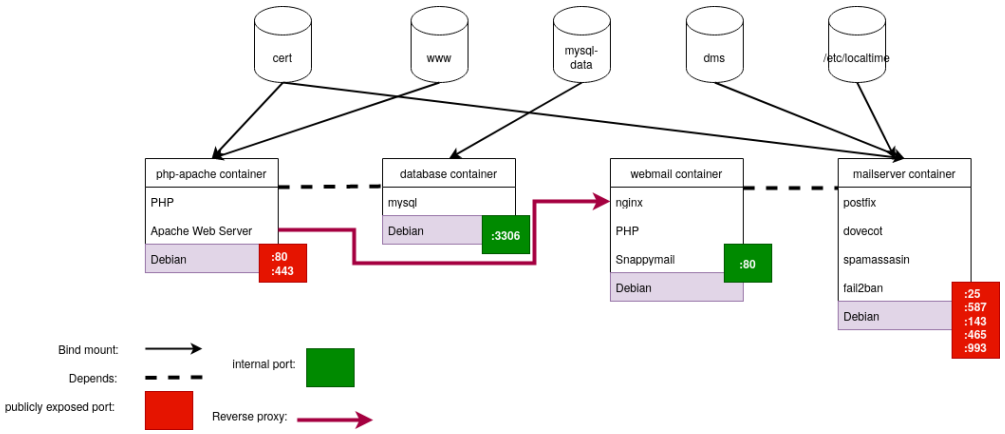
3.4 Kebutuhan Output

Kebutuhan output yang dibutuhkan adalah aplikasi web yang di-*deploy* ke server, dan email yang masuk ke inbox Gmail.

3.5 Prosedur Pengumpulan Data

Untuk data aplikasi yang akan di-*deploy* ke dalam LAMP dibuat oleh Divisi Backend dan Divisi Frontend, sedangkan untuk email diperoleh dari pengguna yang mengirim atau menerima email.

3.6 Analisis dan Rancangan Sistem



Gambar 3.1: Rancangan susunan docker *container*

Sistem yang akan dibuat pada penelitian ini akan terdiri dari empat container seperti pada gambar 3.1, yang memiliki fungsinya masing-masing. Keempat container tersebut dibuat dengan menggunakan basis image debian.

Container "php-apache" digunakan untuk me-*hosting* aplikasi dan/atau halaman web, selain itu juga bertindak sebagai reverse proxy bagi container "webmail". Container ini berisi program Apache Web Server dan PHP interpreter. Container ini akan melakukan *bind mount* ke direktori "www" yang berisi file yang akan di-*host* dan file "certs" yang berisi *certificate* yang digunakan untuk https. Untuk mengakomodasi untuk aplikasi yang menggunakan database MySQL, maka container "php-apache" bergantung dengan container "database", hal ini untuk memastikan

bahwa container “database” harus berjalan jika container “php-apache” dijalankan. Karena web server di container ini yang akan diakses oleh browser, maka port 80 untuk http dan port 443 harus di-*expose* sehingga bisa diakses dari luar.

Container “database” digunakan untuk menjalankan layanan MySQL. Direktori “mysql-data” yang berisi file-file dari database akan di-*mount* ke container ini. Port 3306 pada container ini hanya bisa diakses oleh jaringan internal docker yang dibuat oleh docker secara otomatis.

Container “webmail” berfungsi sebagai Web Client dari email server yang dibuat, sehingga pengguna email dapat membuka akun email melalui browser. Aplikasi Snappymail, yaitu aplikasi email webclient memerlukan PHP dan nginx untuk menjalankannya. Port yang dibuka adalah port 80. Port yang dibuka tersebut merupakan port di jaringan internal docker. Karena web client akan diakses melalui reverse proxy dari container “php-apache”, container ini tidak perlu mengekspose port secara publik, selain itu https juga sudah ditangani oleh container “php-apache” juga. Untuk container “php-apache” sendiri bisa mengakses container “webmail” karena berada di jaringan internal docker yang sama.

Container “mailserver” berfungsi untuk menerima dan mengirimkan email. Container ini memerlukan akses ke direktori “cert” untuk mengakses *certificate* yang digunakan untuk enkripsi TLS pada imap dan smtp, akses ke direktori “dms” untuk menyimpan email, dan direktori “etc/localtime” yang berisi informasi mengenai zona waktu. Container ini menjalankan layanan postfix untuk mengirimkan dan menerima email dari email server lain. Dovecot sebagai server imap, sehingga inbox pengguna dapat diakses melalui protokol imap. Spamassasin untuk menyaring email yang masuk dari email spam. Dan Fail2ban untuk memblokir IP yang mencoba melakukan *brute force* terhadap server. Container ini juga mengekspos port ke publik, yaitu:

1. 25 : digunakan untuk mengirimkan dan menerima email ke email server lain. Serta menerima email dari email client pengguna tanpa TLS yang akan diteruskan ke server email tujuan.
2. 587 : digunakan untuk menerima email dari email client pengguna dengan enkripsi startTLS, yang akan diteruskan ke server email tujuan.
3. 143 : digunakan untuk email client mengakses inbox dengan enkripsi startTLS.
4. 465 : sama dengan port 587 tetapi menggunakan enkripsi TLS.
5. 993 : sama dengan port 143 tetapi menggunakan enkripsi startTLS.

BAB 4

IMPLEMENTASI DAN PEMBAHASAN

4.1 Implementasi dan Uji Coba Sistem

4.1.1 Implementasi

1. Konfigurasi domain

Hal pertama yang perlu dilakukan adalah membuat konfigurasi domain, pada penelitian ini digunakan menggunakan cloudflare.

Type ▲	Name	Content
A	semriwing.my.id	103.160.95.96
CNAME	autoconfig	mail.semriwing.my.id
CNAME	autodiscover	mail.semriwing.my.id
CNAME	mail	semriwing.my.id
CNAME	*	semriwing.my.id
CNAME	www	semriwing.my.id
MX	semriwing.my.id	mail.semriwing.my.id 10
TXT	_dmarc	v=DMARC1; p=quarantine; rua=mail...
TXT	semriwing.my.id	v=spf1 mx ~all

Gambar 4.1: Konfigurasi domain

2. Instalasi Docker

Pada penelitian ini dilakukan instalasi Docker melalui repository dari Docker. Docker sebenarnya sudah tersedia di repository resmi dari Ubuntu server, tetapi Docker tersebut bukan versi yang terbaru.

Tabel 4.1: Langkah instalasi Docker

Perintah	**Keterangan**
`sudo apt-get remove docker docker-engine docker.io containerd runc`	Uninstall docker bawaan dari repository ubuntu
`sudo apt-get update`	Memperbarui data repository
`sudo apt-get install ca-certificates curl gnupg lsb-release`	Menginstall tools yang dibutuhkan untuk menambahkan repository
`sudo mkdir -p /etc/apt/keyrings`	Membuat direktori tempat menyimpan keyring
`curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg`	Mengunduh keyring dan menambahkannya ke sistem

Perintah	Keterangan
`echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null`	Menambahkan url dari repository ke sistem
`sudo apt-get update`	Memperbarui data repository
`sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin`	Mengunduh dan menginstall docker

Setelah proses instalasi docker selesai, docker bisa dicoba dengan menjalankan perintah `docker`.

3. Menyusun Docker Compose

Langkah selanjutnya adalah membuat direktori baru, kemudian buat file docker-compose.yml.

```
version: '3'
services:
  php-apache:
    build:
      context: ./php-apache
    restart: "always"
    ports:
      - 80:80
      - 443:443
    volumes:
      - ./cert/semriwing.my.id:/var/imported/ssl/
      - ./www:/var/www/html:z
    depends_on:
      - 'database'

  database:
    image: mysql:8.0
    restart: "always"
    volumes:
      - ./mysql:/var/lib/mysql
    environment:
      TZ: ${TZ}
      MYSQL_ALLOW_EMPTY_PASSWORD: 'no'
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_DATABASE: 'testdb'

```

Gambar 4.2: Isi dari file docker-compose.yml

```
29   mailserver:
30     image:
     ↪ docker.io/mailserver/docker-mailserver:latest
31     container_name: mailserver
32     restart: "always"
33     hostname: mail
34     domainname: semriwing.my.id
35     ports:
36       - "25:25"
37       - "587:587"
38       - "143:143"
39       - "465:465"
40       - "993:993"
41     volumes:
42       - ./dms/mail-data/:/var/mail/
43       - ./dms/mail-state/:/var/mail-state/
44       - ./dms/mail-logs/:/var/log/mail/
45       - ./dms/config/:/tmp/docker-mailserver/
46       -
     ↪ ./cert/semriwing.my.id/:/tmp/dms/custom-certs/:ro
47       - /etc/localtime:/etc/localtime:ro
48     environment:
49       - ENABLE_FAIL2BAN=1
50       - SSL_TYPE=manual
51       -
     ↪ SSL_CERT_PATH=/tmp/dms/custom-certs/fullchain.pem
52       - SSL_KEY_PATH=/tmp/dms/custom-certs/privkey.pem
53       - ONE_DIR=1
54       - ENABLE_POSTGREY=0
55       - ENABLE_CLAMAV=0
56       - ENABLE_SPAMASSASSIN=1
57     cap_add:
58       - NET_ADMIN # For Fail2Ban to work
59
60   webmail:
61     image: docker.io/kouinkouin/snappymail:latest
62     restart: "always"
63     depends_on:
64       - 'mailserver'
65
```

Gambar 4.3: Isi dari file docker-compose.yml (lanjutan)

4. Menjalankan Service

Setelah selesai membuat konfigurasi dari docker compose, langkah selanjutnya adalah menjalankan container-container yang ada di dalam docker compose tersebut dengan menggunakan perintah `docker compose up -d`, untuk argumen "-d" yang ada bertujuan untuk menjalankan container di *background*.

5. Membuat Akun

Pembuatan akun email diperlukan script yang didapat dari repository github docker-mailserver.

Tabel 4.2: Langkah pembuatan akun email

Perintah	**Keterangan**
`wget https://raw.githubusercontent.com/docker-mailserver/docker-mailserver/master/setup.sh`	Mengunduh script dari github
`chmod a+x ./setup.sh`	Menambahkan *permission* untuk menjalankan script
`./setup.sh email add admin@semriwing.my.id 1234`	Membuat akun email "admin" di domain "semriwing.my.id" dengan password "1234"

6. Membuat key DKIM

```
v=DKIM1;k=rsa;p=MAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAA+0000000000000000000
  0000000000000000000000000
  0000000000000000000000000
  +73rT/5opnRceqQf1qndnMZfkb
  /0/YciMKNQmigj9IGwKypj6CoI
  r1s46jRGy4Ws7LQIDAQAB
```

Gambar 4.4: Contoh Key DKIM

Type ▲	Name	Content
A	semriwing.my.id	103.160.95.96
CNAME	autoconfig	mail.semriwing.my.id
CNAME	autodiscover	mail.semriwing.my.id
CNAME	mail	semriwing.my.id
CNAME	*	semriwing.my.id
CNAME	www	semriwing.my.id
MX	semriwing.my.id	mail.semriwing.my.id 10
TXT	_dmarc	v=DMARC1; p=quarantine; rua=mail...
TXT	mail._domainkey	v=DKIM1; h=sha256; k=rsa; p=MIICI...
TXT	semriwing.my.id	v=spf1 mx ~all

Gambar 4.5: Domain setelah ditambahkan subdomain DKIM

Langkah selanjutnya adalah membuat DKIM supaya email yang dikirim dari email server yang dibuat tidak ditandai sebagai spam. Pembuatan key DKIM diharuskan untuk sudah membuat minimal satu akun email dari domain. Perintah untuk membuat key dkim adalah `./setup.sh config dkim`. Setelah itu key dkim bisa disalin dari direktori yang di-*mount* ke dalam container, pada penelitian ini yaitu `dms/config/opendkim/keys/example.com/mail.txt`, isi dari file tersebut seperti pada gambar 4.4. Isi dari file tersebut disalin dan di taruh ke konfigurasi sub domain “mail._domainkey” seperti pada

gambar 4.5.

4.1.2 Pengujian

Proses pengujian dilakukan untuk mengevaluasi server yang sudah dibuat sudah bisa berjalan sesuai dengan yang diharapkan. Pengujian pada penelitian ini untuk LAMP dilakukan dengan cara membuka halaman utama dari domain, jika aplikasi yang dijalankan berjalan dengan sesuai berarti implementasi LAMP sudah berhasil.

Untuk prosedur pengujian email, dilakukan dengan cara mengirimkan email ke akun Gmail. Jika email bisa melalui SPF, DKIM, dan DMARC check, maka implementasi email berhasil.

4.2 Pembahasan

4.2.1 Docker Compose

Merujuk pada gambar 4.2 di baris nomor pertama terdapat syntax “version”, syntax ini berfungsi untuk menentukan file konfigurasi docker compose tersebut ditulis dengan menggunakan file format versi keberapa. Versi docker compose tidak wajib tapi dianjurkan untuk disertakan. Hal tersebut dilakukan untuk kompatibilitas dengan versi lama.

Pada baris ke-7 di gambar 4.2 terdapat syntax “port”. Disini ditentukan port yang akan dibuka ke publik. Pada nomor port yang ditentukan tersebut terdapat dua port yang ditulis seperti berikut “80:80”. Nomor port sebelum titik dua (:) merupakan port milik host, sedangkan port setelahnya merupakan milik container, jadi port 80 milik host akan disambungkan ke port 80 milik container. Sesuai dengan rancangan di gambar 3.1 port 80 dan 443 dieksps sehingga dapat diakses oleh publik.

Pada baris ke-10 di gambar 4.2 terdapat syntax “volumes”. Syntax ini digunakan untuk menentukan direktori atau berkas yang akan di-*mount* ke dalam kontainer.

Seperti pada gambar 3.1, direktori “cert” dan “www” di-*mount* ke kontainer “php-apache”.

Selanjutnya pada gambar 3.1 kontainer “php-apache” bergantung pada kontainer database, oleh karena itu, pada baris nomor 13 di gambar 4.2 terdapat syntax “depends_on”. Dengan menentukan syntax tersebut maka kontainer database akan otomatis dijalankan jika kontainer “php-apache” berjalan.

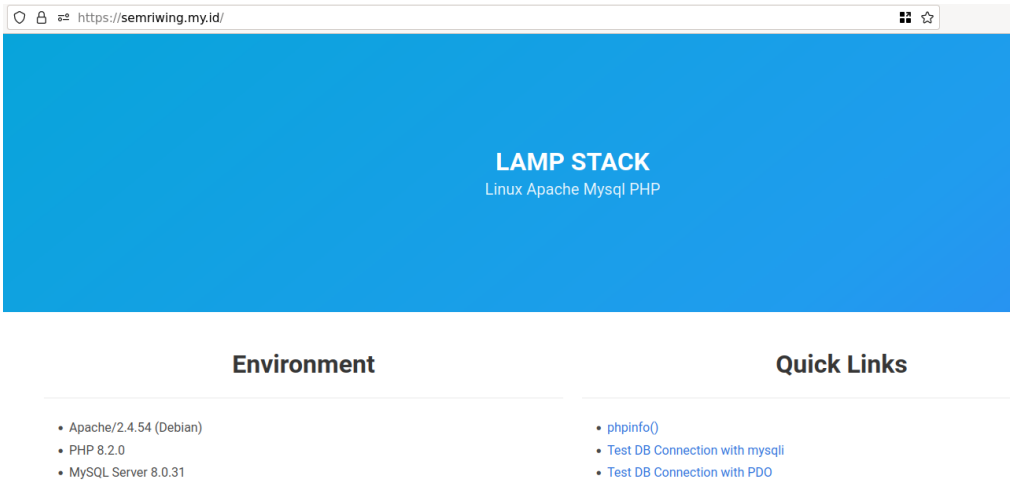
Kontainer database di gambar 4.2 terdapat syntax “environment” di baris 21. Syntax ini digunakan untuk menentukan variabel yang akan dipakai untuk mengatur opsi yang ada di kontainer. Untuk opsi yang bisa dipakai bisa merujuk ke halaman docker hub dari image docker yang dipakai. Pada gambar 3.1, kontainer database memiliki port internal 3306, tetapi port tidak ditentukan di syntax “port”. Hal ini dikarenakan syntax “port” hanya diperlukan jika akan membuka port secara publik. Sedangkan untuk port internal ditentukan di Dockerfile. Pada image yang digunakan oleh kontainer database, port 3306 sudah dibuka secara *default*.

4.2.2 LAMP

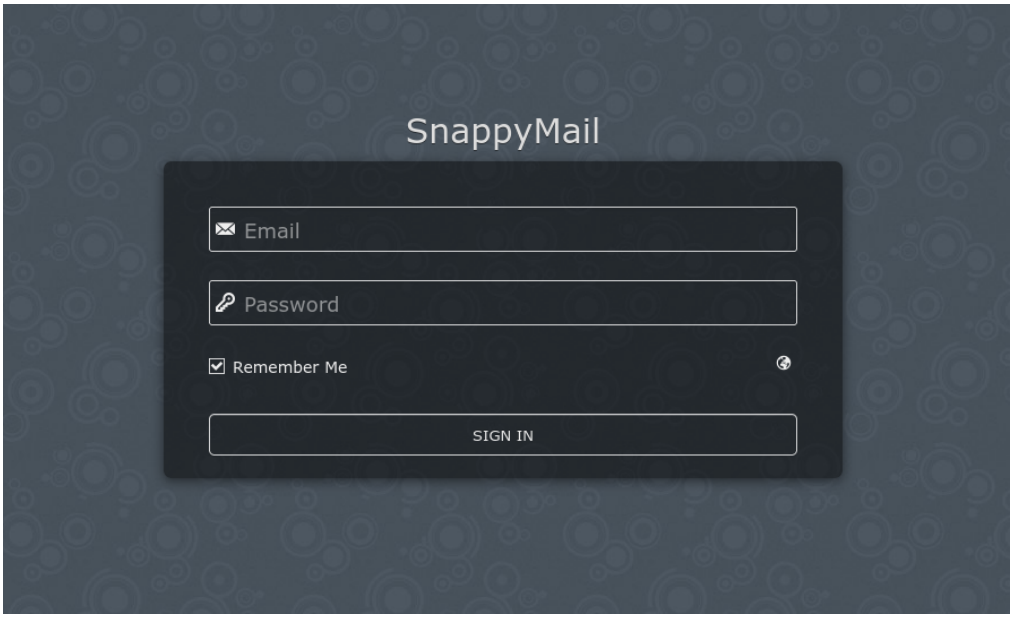
LAMP *stack* yang dipasang di server berhasil berjalan dengan baik seperti yang dilihat pada gambar 4.6. Apache dapat dengan baik menampilkan halaman index. Untuk aplikasi yang dibuat oleh Divisi Backend dan Divisi Frontend yang dideploy dengan menggunakan LAMP *stack* tersebut akan dilakukan pengujian kembali.

4.2.3 Halaman Login Webclient

Gambar 4.7 adalah tampilan halaman login dari webmail snappymail. Pengguna dapat mengakses langsung akun email melalui browser. Pengguna dapat mencentang remember me, sehingga tidak perlu memasukkan password setiap saat.



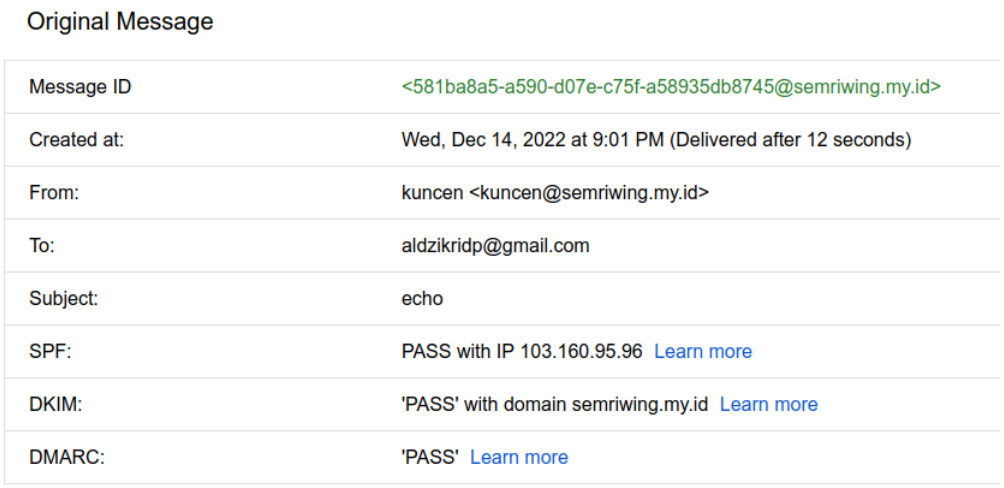
Gambar 4.6: Halaman utama server



Gambar 4.7: Halaman Login Email

4.2.4 Mengirimkan Email

Syarat utama untuk email server yang dibuat adalah dapat mengirimkan email ke akun email Gmail. Hal ini dikarenakan Gmail adalah salah satu layanan penyedia email yang dipakai oleh banyak orang sehingga email server yang dibuat bisa di-


Original Message

Message ID	<581ba8a5-a590-d07e-c75f-a58935db8745@semriwing.my.id>
Created at:	Wed, Dec 14, 2022 at 9:01 PM (Delivered after 12 seconds)
From:	kuncen <kuncen@semriwing.my.id>
To:	aldzikridp@gmail.com
Subject:	echo
SPF:	PASS with IP 103.160.95.96 Learn more
DKIM:	'PASS' with domain semriwing.my.id Learn more
DMARC:	'PASS' Learn more

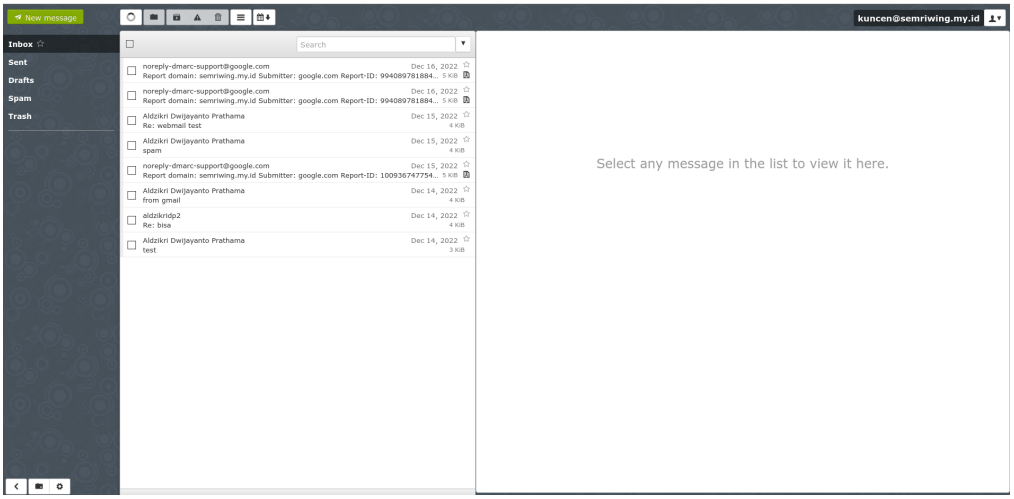
Gambar 4.8: Email yang dikirimkan berhasil melalui cek

pastikan bisa menjangkau banyak orang. Selain itu Gmail juga menerapkan metode verifikasi email pencegah spam secara lengkap. Oleh karena itu dengan melakukan pengujian terhadap email Gmail, email server yang sudah dibuat bisa dianggap sudah memenuhi standar jika berhasil masuk ke inbox Gmail.

Pada server email yang dibangun di penelitian ini, email yang dikirimkan oleh server ini berhasil melewati ketiga autentikasi di atas, nampak pada gambar 4.8.

4.2.5 Menerima Email

Gambar 4.9 merupakan tampilan dari inbox email server yang dibuat. Email yang dikirimkan dari akun Gmail berhasil masuk ke email server.



Gambar 4.9: Inbox Email

BAB 5

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil analisa, implementasi, dan hasil uji coba server yang telah dibangun dalam penelitian ini dapat disimpulkan:

1. Berhasil menghasilkan server LAMP dan server Email menggunakan Docker.
2. Berhasil mengirimkan Email ke Gmail tanpa ditandai sebagi spam.
3. Dengan menggunakan Docker, backup menjadi lebih mudah karena data-data yang dibutuhkan dapat disimpan di satu direktori.
4. Replikasi mudah dilakukan karena dapat menyalin konfigurasi Docker, dan menjalankan satu perintah.

5.2 Saran

Dari penelitian ini memiliki banyak kekurangan yang perlu diperbaiki. Adapun saran yang diberikan untuk mengembangkan penelitian kedepannya adalah sebagai berikut:

1. Menggunakan basis sistem operasi yang ringan untuk docker image, seperti contohnya Alpine linux. Sehingga proses pull dan push image lebih cepat.
2. Menambahkan *reverse* DNS pada server, sehingga email yang dikirimkan dari email server lebih terpercaya dan tidak ditandai sebagai spam oleh penyedia email lain.

DAFTAR PUSTAKA

Amazon (2023). *What Is A Lamp Stack?* URL: https://aws.amazon.com/what-is/lamp-stack/. 25 Januari 2023.

Cloudflare (2023). *What is a mail server?* URL: https://www.cloudflare.com/learning/email-security/what-is-a-mail-server/. 25 Januari 2023.

Docker (2023). *Docker overview*. URL: https://docs.docker.com/get-started/overview/. 25 Januari 2023.

Dwiyatno, Saleh (2020). 'Implementasi virtualisasi server berbasis docker container'. *PROSISKO: Jurnal Pengembangan Riset dan Observasi Sistem Komputer* 7.2, pp. 165–175.

Furnama, Reza (2022). 'Implementasi Arsitektur Microservices Pada Sistem Backend Pembayaran Terintegrasi Menggunakan Docker'. Universitas Islam Negeri Sultan Syarif Kasim Riau.

Kitterman, Scott (Apr. 2014). *Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1*. RFC 7208. DOI: 10.17487/RFC7208.

Kristiawan, Kiki Yuniar, Henni Endah Wahanani dan Mohammad Idhom (2022). 'Implementasi Reverse Proxy Pada Hosting Web Server di Docker Container'. *Jurnal Informatika dan Sistem Informasi* 3.1, pp. 1–12.

Kucherawy, Murray, Dave Crocker dan Tony Hansen (Sept. 2011). *DomainKeys Identified Mail (DKIM) Signatures*. RFC 6376. DOI: 10.17487/RFC6376.

Kucherawy, Murray dan Elizabeth Zwicky (Mar. 2015). *Domain-based Message Authentication, Reporting, and Conformance (DMARC)*. RFC 7489. DOI: 10.17487/RFC7489.

Megantara, Rama Aria et al. (2021). 'IMPLEMENTASI DOCKER DALAM MEMBANTU VIRTUAL LAB PADA UNIVERSITAS'. *Science and Engineering National Seminar*. Vol. 6. 1, pp. 636–639.

Parinda, Siti Annisa et al. (2023). 'PENTINGNYA INOVASI DAN PEMANFAATAN TEKNOLOGI DALAM KEBERAGAMAN DUNIA BISNIS'. *Majalah Ilmiah Inspiratif* 9.16.

Prasetyo, Stefanus Eko dan Ardyansyah Wijaya (2021). 'Analisa dan Implementasi Microservice pada Container Menggunakan Docker'. *CoMBInES-Conference on Management, Business, Innovation, Education and Social Sciences*. Vol. 1. 1, pp. 557–564.

Rosenbaum, Rich (May 1993). *Using the Domain Name System To Store Arbitrary String Attributes*. RFC 1464. DOI: 10.17487/RFC1464.

Saputra, Anggiawan Yus dan Ali Akbar Rismayadi (2021). 'IMPLEMENTASI CLOUD STORAGE MENGGUNAKAN OWNCLUOD DAN DOCKER'. *eProsiding Teknik Informatika (PROTEKTIF)* 2.1, pp. 201–206.

Shah, Jay dan Dushyant Dubaria (2019). ‘Building modern clouds: using docker, kubernetes & Google cloud platform’. *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, pp. 0184–0189.

Wijayanto, Danang (2018). ‘Tehnologi Platfrom Virtualisasi Untuk Aplikasi Lowongan Kerja’. STMIK Akakom Yogyakarta.

Lampiran

Hasil Putusan Sidang

KEPUTUSAN HASIL UJIAN PENDADARAN		
Sesuai dengan hasil sidang pendadaran pada tanggal	21 Februari 2023	maka
Nama Mahasiswa	Aldzikri Dwijayanto Prathama	
NIM / Program Studi	195410189 / Informatika	
Jenjang	S1	
	dinyatakan	LULUS
Ketua Penguji	M. Guntara, Ir., M.T.	

Catatan		
Hari, tanggal	Selasa, 21 Februari 2023	
Waktu	13.00	
Nama	Aldzikri Dwijayanto Prathama	
No. Mahasiswa / Jurusan	195410189 / Informatika	
Hal yang harus diperbaiki		Pemberi Catatan
- kesimpulan dibuat poin-poin. - lihat catatan (comment) di naskah		Guntara
Cek catatan saya di Naskah		Adi K.