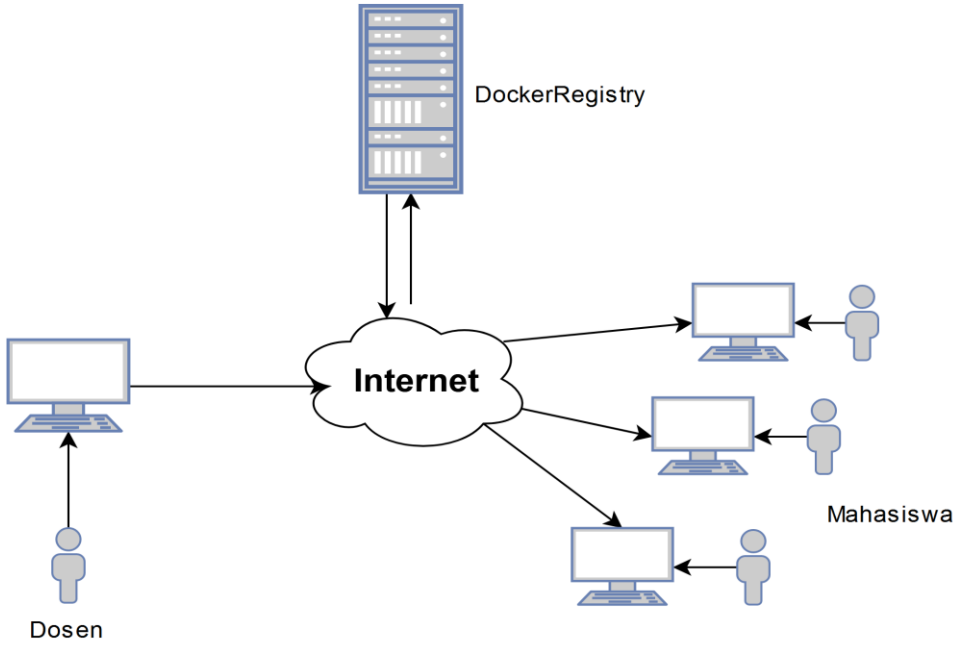
BAB 4 PERANCANGAN

Pada bab ini dijelaskan perancangan sistem yang dibangun dalam pengerjaan tugas akhir. Bagian ini dijelaskan lebih detail mengenai konsep implementasi laboratorium virtual menggunakan *docker* untuk praktikum jaringan komputer dasar. Konfigurasi *dockerfile*, membuat *container* untuk praktikum serta pengujian performasi dan analisis hasil.

4.1 Perancangan Sistem

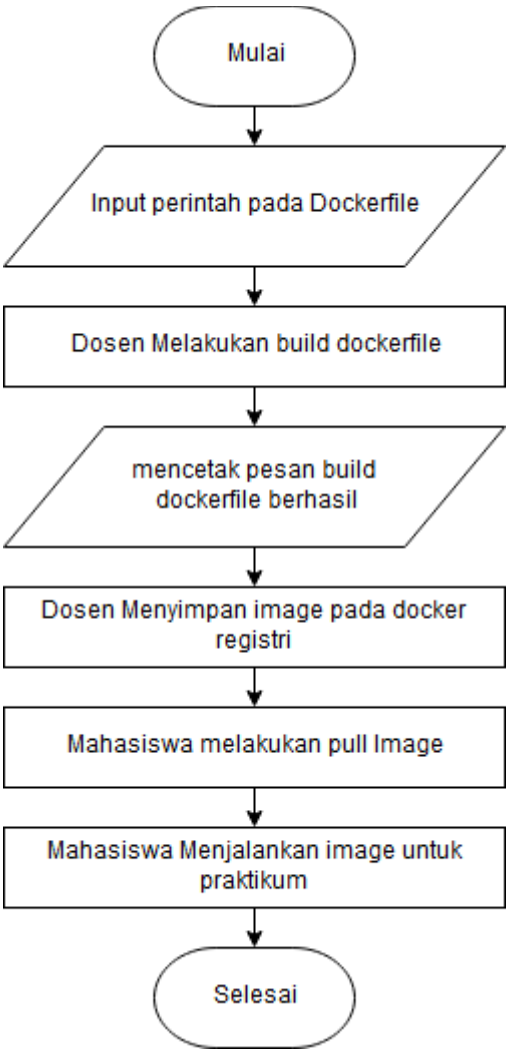


Gambar 4.1 Perancangan Sistem

Gambar 4.1 menunjukan konsep perancangan sistem pada penelitian ini. Proses dilakukan dengan beberapa komponen utama yaitu dosen atau pengajar dan mahasiswa. Sistem ini juga membutuhkan komponen pendukung yaitu tempat penyimpanan secara *cloud* pada docker *registry*. Syitem ini membutuhkan koneksi internet agar masing-masing komponen saling terhubung. Proses diawali dengan dosen atau pengajar membuat materi untuk praktikum jaringan komputer. Materi praktikum dibuat dengan melakukan konfigurasi pada *dockerfile*. Selanjutnya *dockerfile* di-*build* hingga menjadi *image*. Kemudian *image* di *push* atau di *upload* ke *repository dockerhub* dengan menggunakan internet. Untuk menyimpan *image* pada *dockerhub* diperlukan akun agar dapat membuat *repository*. Setelah *image* tersimpan, mahasiswa dapat *pull* atau mengunduh *image* dari *docker registry*. Untuk melakukan pengaksesan terhadapat image pada docker registry mahasiswa dapat menggunakan terminal pada linux. Mahasiswa dapat *pull* atau *download image* menggunakan perintah *command line* pada terminal.

4.2 Alur Kerja Sistem

Pada bab ini dijelaskan perancangan sistem dari tahap konfigurasi *dockerfile*, pembuatan *container* praktikum hingga hasil pengujian dan kesimpulan yang dilakukan pada sistem. Dibawah ini adalah *flowchart* perancangan sistem laboratorium virtual mandiri menggunakan *docker* untuk praktikum jaringan komputer dasar:



Gambar 4.2 Diagram Alir Perancangan Sistem

Pada gambar 4.2 dapat dilihat bahwa di tahap perancangan ini diawali dengan konfigurasi *dockerfile*. Untuk penjelasan lebih detail mengenail gambar 4.2 akan dibahas sebagai berikut:

1. Input perintah *dockerfile*. Konfigurasi dilakukan dengan menuliskan sekumpulan perintah atau *script* yang digunakan untuk membuat materi praktikum. *Script* tersebut dituliskan pada *nano* yang merupakan editor pada linux, kemudian file disimpan dengan nama *Dockerfile*.

2. Melakukan *build* dockerfile. Pada tahap ini *dockerfile* telah dikonfigurasi dengan isi *script* sesuai keperluan praktikum. *Build* dilakukan untuk membuat *dockerfile* menjadi *image*.
3. Proses *build dockerfile* telah selesai ketika pada terminal tercetak *output* bahwa proses *build* yang dilakukan telah berhasil.
4. Menyimpan *image* pada *docker registry.* Tahap ini dilakukan saat *image* sudah terbuat kemudian di *push* pada *docker registry. Image* yang telah di *push* akan tersimpan secara *cloud.* Pada penelitian disimpan tiga *image* sesuai materi praktikum pada modul praktikum jaringan komputer dasar.
5. Mahasiswa melakukan *pull image* pada masing-masing komputernya untuk melakukan praktikum sesuai bab yang telah di pelajari. Mahasiswa melakukan *pull* dengan perintah sederhana pada terminal linux.
6. Menjalankan *image* praktikum. Tahapan ini dipilih salah satu *image* sesuai praktikum yang sedang dilakukan kemudian di *pull* pada masing-masing laptop. *Image* praktikum dapat dijalankan melalui *terminal* linux untuk mengakses *image*.

4.2.1 Alur Kerja Sistem Untuk Dosen Atau Pengajar

Dosen atu pengajar merupakan salah satu aktor yang diperlukan sistem untuk membuat materi untuk laboratorium virtual komputer yang digunakan sebagai praktikum. Berikut ini penjelasan lebih rinci mengenai alur kerja system pada dosen atau pengajar:

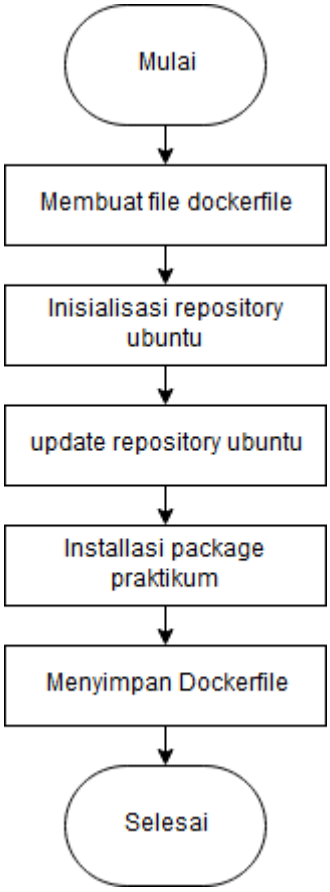
4.2.1.1 Input Perintah *Dockerfile*

Tahap awal untuk membuat materi praktikum untuk laboratorium vitrtual mandiri menggunakan *docker* adalah dengan mengkonfigurasi *dockerfile*. Dalam pembuatan *dockerfile* memiliki beberapa tahapan seperti berikut:

1. *Dockerfile* dibuat melalui editor pada linux yaitu *nano*. *Dockerfile* pada dasarnya berisi sekumpulan *script* untuk merancang materi praktikum
2. Melakukan inisialisasi pada *repository* ubuntu agar *dockerfile* dapat terhubung dengan *repository* ubuntu untuk pengambilan *package-package* yang diperlukan.
3. Melakukan *update* pada *repository* ubuntu. Hal ini dilakukan agar aplikasi-aplikasi yang di-*download* terupdate serta bug dapat teratasi.
4. Tahapan selanjutnya adalah melakukan instalasi aplikasi yang diperlukan untuk kebutuhan praktikum. Pada tahap ini *script* di sesuaikan dengan praktikum yang akan dibuat. Pada penelitian ini ada 3 materi praktikum yang akan dibuat yaitu *packet capturing*, pemrograman *socket*, *protocol layer transport*.

5. Tahap terakhir yaitu menyimpan konfigurasi yang telah dibuat. *file* disimpan dengan nama *Dockerfile* tanpa diberi tipe *file* agar dapat di *build* menjadi *image*.

   Berikut ini merupakan *flowchart* dari konfigurasi dockerfile:



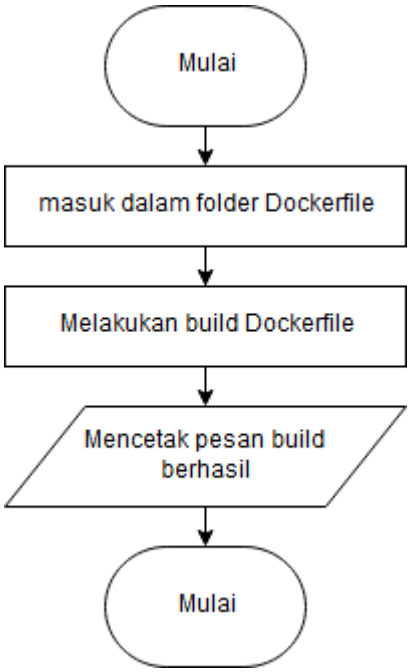
Gambar 4.3 *Flowchart* Konfigurasi *Dockerfile*

4.2.1.2 Melakukan *Build Dockerfile*

Tahap selanjutnya adalah *build image*. Untuk *build image* hanya dilakukan setelah konfigurasi materi pada *dockerfile* selesai dibuat. *build* ini dapat dilakukan oleh dosen atau pengajar. *Build* bertujuan merubah *dockerfile* menjadi *image*. Berikut ini alur untuk melakukan pembuatan *image*:

1. Langkah awal untuk melakukan *build file* dapat dilakukan dengan masuk pada folder yang berisi *dockerfile*.
2. Selanjutnya adalah *build file* dengan nama *Dockerfile* menggunakan perintah *docker build*. Pada saat *build file*, terminal akan menampilkan setiap proses file yang di-*build* mulai dari inisialisai *repository* hingga proses instalasi aplikasi yang dibutuhkan untuk praktikum
3. Tahap terakhir adalah proses *build* telah selesai. *Dockerfile* telah menjadi *image*. *Build* sukses dapat dilihat saat terminal menampilkan *output build* sukse dan menampilkan *image ID*. *Image ID* biasanya berjumalah 12 karakter angka dan huruf secara acak.

Flowchart dari penjelasan diatas dapat dilihat pada gambar 4.4 dibawah ini:



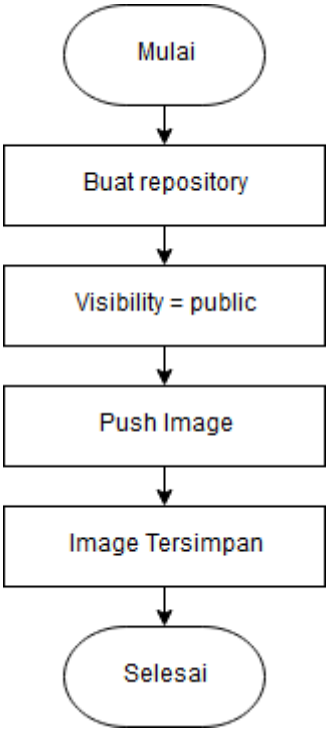
Gambar 4.4 *Flowchart build docker*

4.2.1.3 Menyimpan Image

Penyimpanan *image* dilakukan secara *cloud* pada *docker registry*. Untuk melakukan penyimpanan (*push*) *image* diperlukan registrasi pada *dockerhub*, kemudian setelah memiliki akun maka dapat membuat *repository* sendiri untuk menyimpan *image*. Nama *repository* yang dibuat merupakan nama yang sama dengan nama *image* yang akan di *push*. Berikut ini alur menyimpan *image*:

1. Buat *repository*. Seperti yang telah dijelaskan sebelumnya *repository* dapat dibuat setelah memiliki akun pada *dockerhub*. Pada *dockerhub* ini user dapat secara bebas membuat *repository* yang diinginkan
2. Selanjutanya adalah merubah *visibity* menjadi publik agar *image* dapat diakses oleh *user* secara publik
3. Melakukan *push image* pada *dockerhub* dengan perintah *docker push* pada *terminal* linux. Nama *image* dan *repository* merupakan nama yang sama untuk menghindari kegagalan *push*.

   Berikut merupakan *flowchart* untuk menyimpan image pada repository:



Gambar 4.5 *Flowchart* Menyimpan *Image*

4.2.2 Alur Kerja Sistem Untuk Mahasiswa

Aktor selanjutnya yang ada pada sistem adalam mahasiswa. Mahasiswa melakukan praktikum sesuai dengan bab praktikum jaringan komputer dasar yang telah dipelajari. Pada sistem yang di rancang ini mahasiswa dapat mengerjakan materi praktikum pada masing-masing komputer dengan cara *pull image* yang berada pada *repository dockerhub*. Berikut ini penjelasan lebih rinci mengenai alur kerja sistem untuk mahasiswa:

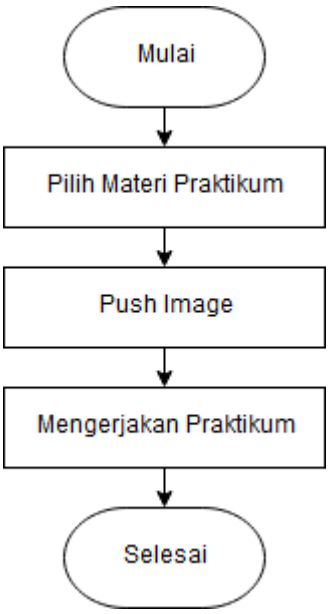
4.2.2.1 Menjalankan Image Praktikum

Image praktikum yang sudah di *pull* dapat dijalankan dengan perintah *docker* pada terminal. *Image* praktikum ini dapat dijalankan pada masing-masing komputer mahasiswa melalui terminal linux. Berikut ini merupakan alur untuk mengerjakan praktikum menggunakan *docker*:

1. Tahap pertama adalah memilih materi praktikum. Pemilihan materi ini sesuai bab praktikum yang telah dilakukan oleh mahasiswa. Nama bab praktikum sesuai dengan nama *image* untuk memudahkan penarikan (*pull*) *image* ke dalam komputer.
2. Tahap selanjutnya melakukan *pull image* sesuai bab praktikum yang berada pada *repository*. Untuk melakukan *pull* diperlukan koneksi internet
3. Tahap terakhir, mengerjakan praktikum dengan menjalankan *image* yang telah di *pull* dengan melakukan *run* pada *image* yang telah di *pull* menggunakan perintah *docker run* nama *image*. Dari *image* yang telah di *run*

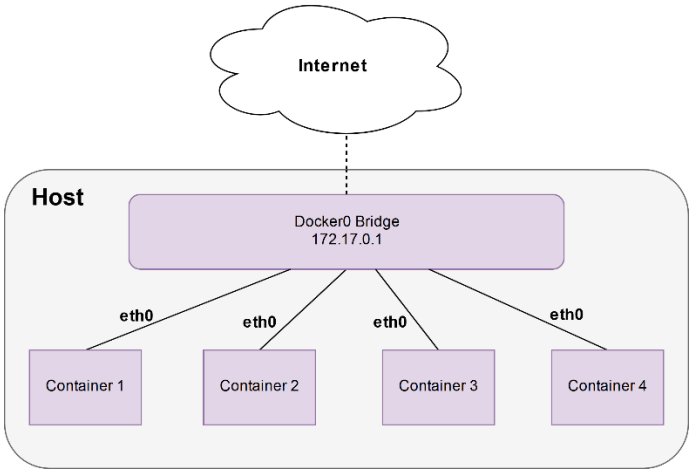
akan membuat *container* baru yang berisi aplikasi serta konfigurasi yang diperlukan materi praktikum.

Berikut ini merupakan *flowchart* menjalankan image praktikum:



Gambar 4.6 *Flowchart* Menjalankan *Image* Praktikum

4.3 *Docker Network*



Gambar 4.7 *Docker Network*

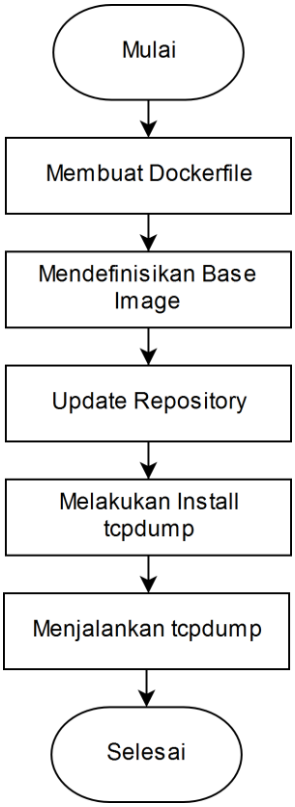
Docker network merupakan bagian yang menjelaskan *container docker* untuk dapat terhubung dengan internet. Untuk memastikan *container docker* dapat terhubung pada internet, *host* dari *container* harus terhubung pada internet. *Host* terhubung pada internet menggunakan jaringan *wireless*. *Container* membutuhkan internet agar dapat melakukan *update* dan instalasi paket yang dibutuhkan untuk praktikum. *Docker container* terhubung pada internet menggunakan *bridge* secara *default*. *Container* terhubung pada *driver network docker0* yang tersedia dalam paket instalasi *docker*. Ketika *container* dijalankan,

container secara otomatis akan terhubung pada *docker0* yang ada pada *host* dengan bridge. Oleh karena itu docker dapat mengakses internet karena terhubung dengan *host*.

4.4 Perancangan Image Praktikum

Image praktikum menyediakan materi yang dirancang sebelumnya sesuai dengan bab praktikum. Pada tugas akhir ini diimplementasikan 3 bab praktikum dalam sebuah *image* yaitu *packet capturing*, pemrograman *socket* dan *protocol layer transport*. Dibawah ini adalah penjelesan lebih detail mengenai perancangan *image* praktikum:

4.4.1 Image *Packet Capturing*



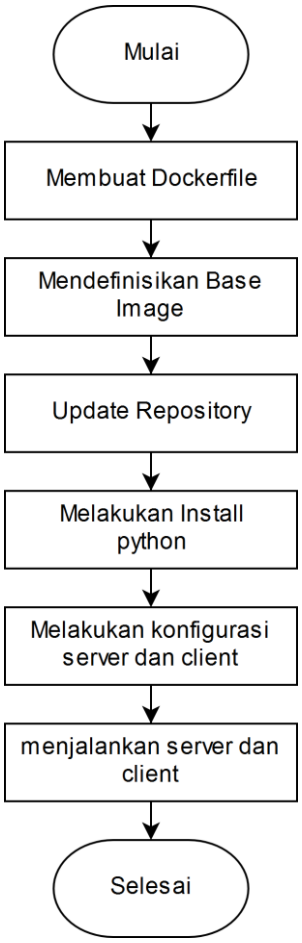
Gambar 4.8 Perancangan *Image Packet Capturing*

Bab pertama pada praktikum jaringan komputer adalah *packet capturing*. Pada praktikum ini melakukan penangkapan *packet* pada jaringan menggunakan aplikasi tcpdump. Berikut ini penjelesan lebih detail mengenai perancangan *image packet capturing*:

1. Membuat *dockerfile*. *Dockerfile* digunakan untuk mendefinisikan *image* yang dibuat.
2. Mendefinisikan sebuah *base image*. *Base image* digunakan sebagai sistem operasi dan *repository* yang akan digunakan pada *image*.

3. Melakukan *update* pada *repository*. *Repository* yang digunakan pada sistem operasi harus diperbarui agar dapat melakukan instalasi paket-paket yang dibutuhkan dalam *image*.
4. Melakukan instalasi *tcpdump*. Tcpdump digunakan sebagai perangkat lunak untuk melakukan praktikum.
5. Menjalankan *tcpdump*. *Tcpdump* digunakan untuk menagkap paket pada jaringan.

4.4.2 Image Pemrograman *Socket*



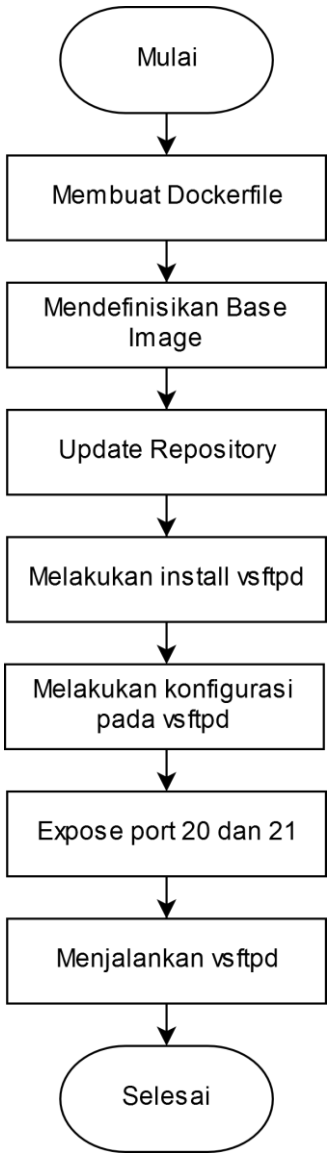
Gambar 4.9 Perancangan *Image* Pemrograman *Socket Server* dan *Client*

Bab kedua pada jaringan komputer adalah pemrograman *socket*. Untuk melakukan pemrogram *socket* dibutuh *server* dan *client* menggunakan aplikasi *python*. Berikut ini merupakan penjelasan mengenai Gambar 4.9:

1. Membuat *dockerfile*. *Dockerfile* digunakan untuk mendefinisikan *image* yang dibuat.
2. Mendefinisikan sebuah *base image*. *Base image* digunakan sebagai sistem operasi dan *repository* yang akan digunakan pada *image*.

3. Melakukan *update* pada *repository*. *Repository* yang digunakan pada sistem operasi harus diperbarui agar dapat melakukan instalasi paket-paket yang dibutuhkan dalam *image*.
4. Melakukan instalasi *python*. *Python* digunakan sebagai perangkat lunak untuk melakukan praktikum.
5. Melakukan konfigurasi *server* dan *client* pada masing-masing *container*. Bagian *server* dikonfigurasi untuk menerima koneksi dari *client*. Sedangkan *client* dikonfigurasi untuk mengirimkan pesan pada *server* melalui *input*.
6. Menjalankan *server* dan *client*. *Server* dan *client* harus dijalankan agar dapat melakukan komunikasi pada praktikum.

4.4.3 Image Protocol Layer Transport



Gambar 4.10 Perancangan *Image Protocol Layer Transport*

Bab ketiga pada jaringan komputer adalah *protocol layer transport*. Untuk melakukan praktikum ini dibutuhkan perangkat lunak *vsftpd* sebagai *server*. Berikut ini merupakan penjelasan mengenai gambar *flowchart* 4.10:

1. Membuat *dockerfile*. *Dockerfile* digunakan untuk mendefinisikan *container* yang aku dibuat.
2. Mendefinisikan sebuah *base image*. *Base image* digunakan sebagai sistem operasi dan *repository* yang akan digunakan pada *image*.
3. Melakukan *update* pada *repository*. *Repository* yang digunakan pada sistem operasi harus diperbarui agar dapat melakukan instalasi paket-paket yang dibutuhkan dalam *image*.
4. Melakukan instalasi *vsftpd*. *Vsftpd* digunakan sebagai perangkat lunak yang dibutuhkan dalam melakukan praktikum.
5. Melakukan konfigurasi pada *file vsftpd.conf*. Konfigurasi *vsftpd* bertujuan agar dapat perangkat lunak *vsftpd* dapat berjalan sesuai kebutuhan.
6. *Expose port* yang digunakan *vsftpd*. *Expose* bertujuan untuk mengaktifkan *port* yang digunakan pada implementasi *vsftpd*.
7. Menjalankan *vsftpd* sesuai dengan konfigurasi yang dilakukan. ftp menggunakan ip sesuai dengan adapter *docker0* pada *host*.