ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - KI141502

RANCANG BANGUN MIDDLEWARE UNTUK MELACAK KONFIGURASI PERANGKAT JARINGAN MENGGUNAKAN GIT

MUHAMMAD FARIS DIDIN ANDIYAR
NRP 05111540000118

Dosen Pembimbing I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D

Dosen Pembimbing II
Bagus Jati Santoso, S.Kom., Ph.D

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2019

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR - KI141502

RANCANG BANGUN MIDDLEWARE UNTUK MELACAK KONFIGURASI PERANGKAT JARINGAN MENGGUNAKAN GIT

MUHAMMAD FARIS DIDIN ANDIYAR
NRP 05111540000118

Dosen Pembimbing I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D

Dosen Pembimbing II
Bagus Jati Santoso, S.Kom., Ph.D

DEPARTEMENT INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2019

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS - KI141502

DESIGN OF MIDDLEWARE FOR TRACK NETWORK DEVICE CONFIGURATION WITH GIT

MUHAMMAD FARIS DIDIN ANDIYAR
NRP 05111540000118

Supervisor I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D

Supervisor II
Bagus Jati Santoso, S.Kom., Ph.D

DEPARTEMENT OF INFORMATICS
Faculty of Information Technology and Communication
Institut Teknologi Sepuluh Nopember
Surabaya, 2019

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

RANCANG BANGUN MIDDLEWARE UNTUK MELACAK KONFIGURASI PERANGKAT JARINGAN MENGGUNAKAN GIT

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Arsitektur Jaringan dan Komputer
Program Studi S1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :
MUHAMMAD FARIS DIDIN ANDIYAR
NRP: 05111540000118

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D .....
NIP: 197708242006041001 (Pembimbing 1)

Bagus Jati Santoso, S.Kom., Ph.D .....
NIP: 198611252018031001 (Pembimbing 2)

SURABAYA
Juni 2019

(Halaman ini sengaja dikosongkan)

RANCANG BANGUN MIDDLEWARE UNTUK MELACAK KONFIGURASI PERANGKAT JARINGAN MENGGUNAKAN GIT

Nama	:	**MUHAMMAD FARIS DIDIN ANDIYAR**
NRP	:	**05111540000118**
Departemen	:	**Informatika FTIK**
Pembimbing I	:	**Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D**
Pembimbing II	:	**Bagus Jati Santoso, S.Kom., Ph.D**

Abstrak

Saat ini infrastruktur jaringan semakin kompleks dan terdiri dari banyak perangkat. Seiring dengan perubahan kebutuhan maka pengaturan dari infrastruktur jaringan juga akan selalu berubah. Perangkat jaringan yang ada pada saat ini hanya mampu menyimpan konfigurasi kedalam file tetapi tidak bisa menyimpan perubahan yang sudah dilakukan. Konfigurasi dari perangkat jaringan akan berubah-ubah sesuai dengan kebutuhan infrastruktur jaringan yang ada. Meskipun selalu berubah, tidak menutup kemungkinan kita ingin melihat atau menggunakan konfigurasi yang sudah lama.

(VCS) Version Control System merupakan cara yang saat ini umum digunakan untuk mencatat setiap perubahan yang ada pada file sehingga kita dapat melacak perubahan yang ada. VCS akan menyimpan setiap perubahan yang ada pada file dan mencatatnya di dalam database repositori dalam bentuk urutan perubahan dari waktu ke waktu. Salah satu VCS yang sekarang banyak digunakan adalah Git.

Dalam tugas akhir ini akan dibuat rancangan sebuah sistem yang memungkinkan untuk membuat versioning dari setiap

konfigurasi perangkat jaringan menggunakan Git. Sistem ini bisa menyimpan catatan perubahan dari file konfigurasi perangkat jaringan ke dalam server. Jika dibutuhkan versi konfigurasi yang lama, konfigurasi bisa diambil dari catatan perubahan yang disimpan di dalam server.

Hasil uji coba menunjukkan middleware dapat digunakan untuk melacak perubahan konfigurasi yang disimpan di dalam repositori. Versi yang diunduh kedalam perangkat jaringan dapat ditentukan oleh middleware. Waktu yang diperlukan untuk merubah versi cukup lama bergantung pada jumlah perubahan yang disimpan. Untuk mengatasi hal itu perlu dioptimalkan cara melihat daftar perubahan yang disimpan.
***Kata-Kunci**: version control system, git, versioning.*

DESIGN OF MIDDLEWARE FOR TRACK NETWORK DEVICE CONFIGURATION WITH GIT

Name	:	**MUHAMMAD FARIS DIDIN ANDIYAR**
NRP	:	**05111540000118**
Department	:	**Informatics FTIK**
Supervisor I	:	**Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D**
Supervisor II	:	**Bagus Jati Santoso, S.Kom., Ph.D**

Abstract

Nowadays network infrastructures are more complex and consist of many devices. The configuration of network infrastructure will change depend on organization's requirements. Current network devices only able to save configuration into file but cannot save the changes of the configuration. Sometime we need to rollback to older version of configuration so we need ability to change the configuration as we want.

(VCS) Version Control System is the common mechanism to save every changes that happen in a single file. With VCS we will able to track the version of a file. VCS wil save the changes of file into repository's database. One of VCS that mostly used is Git.

This final project will make tools that able to make a versioning of network devices configuration with Git. System will able to save changing record of a file configuration into a server. When network devices need older version of configuration network devices can download configuration from the system. The trial results show middleware can be used to track configuration changes that are stored in the repository.

The version downloaded to the network device can be determined by the middleware. The amount of time needed to change the version depends on the number of changes saved. The ammount of time to show all list changes are quiet long, to overcome this middleware need to be optimized on how to see a list of changes that are saved.
***Kata-Kunci**: version control system, git, versioning.*

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **Rancang Bangun *Middleware* untuk Melacak Konfigurasi Perangkat Jaringan Menggunakan Git**. Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis. Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Departemen Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari. Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT atas anugerahnya yang tidak terkira kepada penulis dan Nabi Muhammad SAW.
2. Keluarga penulis yang selalu menyemangati.
3. Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D selaku pembimbing I yang telah membantu, membimbing dan memotivasi penulis mulai dari pengerjaan proposal hingga terselesaikannya Tugas Akhir ini.
4. Bapak Bagus Jati Santoso, S.Kom., Ph.D selaku pembimbing II yang juga telah membantu, membimbing dan memotivasi penulis mulai dari pengerjaan proposal hingga terselesaikannya Tugas Akhir ini.
5. Teman-teman *Administrator* laboratorium AJK.
6. Darlis Herumurti, S.Kom., M.Kom., selaku Kepala Departemen Informatika ITS pada masa pengerjaan Tugas Akhir, Bapak Radityo Anggoro, S.Kom., M.Sc., selaku

koordinator TA dan segenap dosen Departemen Informatika yang telah memberikan ilmu dan pengalamannya.

7. Serta semua pihak yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Juni 2019

Muhammad Faris Didin A.

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I

PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir dan sistematika penulisan.

1.1 Latar Belakang

Di dalam suatu instansi, arsitektur jaringan merupakan bagian yang sangat penting untuk menunjang kinerja dari instansi tersebut. Semakin besar suatu instansi maka arsitektur jaringan disana juga semakin kompleks. Oleh karena itu diperlukan suatu sistem yang mampu mengatur seluruh perangkat jaringan dengan mudah, sehingga administrator jaringan dapat bekerja secara maksimal.

Saat ini perangkat jaringan hanya memiliki *filesystem* untuk penyimpanan konfigurasi. Namun tidak memiliki mekanisme penyimpanan perubahan yang terjadi pada konfigurasi, sehingga kesulitan dalam melacak versi konfigurasi setelah melakukan banyak perubahan konfigurasi pada perangkat. Hal ini juga dialami oleh DPTSI ITS. Ketika administrator jaringan ingin melihat versi konfigurasi pada waktu tertentu maka akan kesulitan karena sulit untuk identifikasi versi perubahan yang ada.

Dalam perkembangan teknologi saat ini banyak terdapat alat untuk melacak perubahan konfigurasi yang disebut VCS(Version Control System) seperti Git, Subversion, dan Bazaar. Untuk menyelesaikan permasalah administrator jaringan dalam melacak perubahan konfigurasi dibutuhkan VCS seperti Git untuk menyimpan perubahan konfigurasi perangkat jaringan. Git merupakan VCS yang umum digunakan oleh pengembang aplikasi[1]. Selain itu dibandingkan *file versioning* lain Git lebih cepat dalam proses penggunaannya[2].

Pada tugas akhir ini akan dibuat sebuah sistem untuk melacak perubahan file konfigurasi perangkat jaringan dalam bentuk aplikasi web yang memanfaatkan Git sebagai penyimpanan perubahan file konfigurasi.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut :

1. Bagaimana merancang *versioning* penyimpanan konfigurasi perangkat jaringan berbasis git?
2. Bagaimana merancang *middleware* protokol penyimpanan konfigurasi perangkat jaringan untuk *versioning* penyimpanan konfigurasi secara transparan?
3. Bagaimana merancang sistem informasi *backend* untuk administrator untuk pengelolaan versi konfigurasi?
4. Bagaimana mengimplementasi sistem *versioning* untuk perangkat jaringan di DPTSI ITS?

1.3 Batasan Masalah

Dari permasalahan yang telah diuraikan di atas, terdapat beberapa batasan masalah pada tugas akhir ini, yaitu:

1. Perangkat jaringan yang digunakan adalah *router* dan *switch*.
2. Perangkat jaringan merupakan produk dari Cisco, Huawei, dan Mikrotik.

1.4 Tujuan

Tujuan pembuatan tugas akhir ini antara lain:

1. Membuat *versioning* konfigurasi perangkat jaringan berbasis git.

2. Membuat *middleware* untuk menjembatani penyimpanan konfigurasi perangkat jaringan.
3. Membuat sistem informasi *backend* untuk pengelolaan versi konfigurasi.
4. Membuat sistem untuk *versioning* konfigurasi perangkat jaringan di DPTSI ITS.

1.5 Manfaat

Manfaat dari pembuatan tugas akhir ini adalah mempermudah melacak versi konfigurasi perangkat jaringan.

1.6 Metodologi

Metodologi yang digunakan dalam pembuatan Tugas Akhir ini adalah sebagai berikut.

1.6.1 Penyusunan Proposal Tugas Akhir

Proposal tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan tugas akhir ini terdiri dari hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah pada tugas akhir, tujuan dari pembuatan tugas akhir dan manfaat dari hasil pembuatan tugas akhir. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Sub bab metodologi berisi penjelasan mengenai tahapan penyusunan tugas akhir mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

1.6.2 Studi Literatur

Pada tahap ini dilakukan pencarian informasi dan referensi mengenai Git dan Python Watchdog untuk mendukung dan memastikan setiap tahap pembuatan tugas akhir sesuai dengan prosedur yang berlaku serta dapat diimplementasikan. Sumber informasi dan referensi bisa didapatkan melalui buku, jurnal, dan internet.

1.6.3 Analisis dan Desain Perangkat Lunak

Pada tahap ini dilakukan analisis dan perancangan terhadap arsitektur tugas akhir yang akan dibuat. Tahap ini merupakan tahap yang paling penting dimana segala bentuk implementasi dapat berjalan dengan baik ketika arsitektur sistem juga baik.

1.6.4 Implementasi Perangkat Lunak

Pada tahap ini dilakukan implementasi atau realisasi dari anilisis dan perancangan arsitektur sistem yang sudah dibuat sebelumnya, sehingga menjadi infrastruktur yang sesuai dengan apa yang direncanakan.

1.6.5 Pengujian dan Evaluasi

Pada tahap ini dilakukan pengujian untuk mengukur performa dari sistem penyimpanan konfigurasi perangkat jaringan menggunakan arsitektur sistem yang telah dibuat. Beberapa performa yang diukur antara lain, kecepatan protokol pengiriman dan ketepatan versi dengan perubahan yang ada. Setela dilakukan ujicoba, maka dilakukan evaluasi terhadap kinerja arsitektur sistem yang telah diimplementasikan dengan tujuan bisa diperbaiki jika ada pengembangan selanjutnya.

1.6.6 Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan buku tugas akhir yang berisi dokumentasi yang mencakup teori, konsep, implementasi dan hasil pengerjaan tugas akhir.

1.7 Sistematika Penulisan

Sistematika penulisan laporan tugas akhir secara garis besar adalah sebagai berikut :

1. Bab I. Pendahuluan
   Bab ini berisi penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi dan sistematika penulisan dari pembuatan tugas akhir.
2. Bab II. Tinjauan Pustaka
   Bab ini berisi kajian teori atau penjelasan metode, algoritma, *library* dan *tools* yang digunakan dalam pembuatan tugas akhir ini. Kajian teori yang dimaksud berisi tentang penjelasan singkat mengenai *Python*, *Flask*, *Gitpython* dan *Python Watchdog*.
3. Bab III. Desain dan Perancangan
   Bab ini berisi mengenai analisis dan perancangan arsitektur sistem yang akan diimplementasikan dalam pembuatan tugas akhir.
4. Bab IV. Implementasi
   Bab ini berisi mengenai implementasi dari arsitektur sistem yang dibuat sebelumnya. Penjelasan berupa kode program dan pengaturan yang digunakan untuk implementasi arsitektur sistem.
5. Bab V. Pengujian dan Evaluasi
   Bab ini berisi tentang tahapan ujicoba terhadap performa arsitektur sistem dan evaluasi terhadap sistem yang dibuat.
6. Bab VI. Penutup
   Bab ini merupakan bab terakhir yang memaparkan

kesimpulan dari hasil pengujian dan evaluasi yang telah dilakukan. Pada bab ini juga terdapat saran yang ditujukan bagi pembaca yang berminat untuk melakukan pengembangan terhadap tugas akhir ini.

7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam tugas akhir.
8. Lampiran
   Dalam lampiran terdapat kode sumber program secara keseluruhan.

BAB II

TINJAUAN PUSTAKA

2.1 Python

Python adalah bahasa pemrograman interpretatif, interaktfi dan berorientasi objek. Python menggabungkan modul, pengecualian, penulisan secara dinamis, tipe data dinamis yang sangat tinggi dan kelas. Python memiliki antarmuka ke banyak *system call* dan pusataka diberbagai sistem dan dapat diperluas ke bahasa pemrograman C atau C++. Python dapat berjalan pada berbagai sistem operasi seperti Unix, Linux, Mac Os dan Windows.

Python adalah bahasa pemrograman tingkat tinggi yang dapat diterapkan pada berbagai masalah. Bahasa ini dilengkapi pustaka yang besar untuk melakukan pemrosesan *string*, protokol internet, rekayasa perangkat lunak dan antarmuka sistem operasi[3].

Dilihat dari kelebihan, bahasa pemrograman Python dapat digunakan dalam pengembangan aplikasi yang kompleks. Pada tugas akhir ini, bahasa pemrograman Python digunakan untuk pembuatan *middleware*.

2.2 Flask

Flask adalah kerangka aplikasi web Python yang ringan. Flask dirancang untuk memulai membuat web dengan cepat dan mudah, dengan kemampuan untuk membuat aplikasi web sampai tingkat yang rumit. Flask dibuat dengan terintegrasi dengan modul Werkzeug dan Jinja. Flask termasuk salah satu kerangka aplikasi web Python yang populer.

Flask didesain tidak memiliki depedensi dan tata letak kerangka aplikasi, dengan demikian pengembang memiliki kebebasan untuk mengatur kerangka aplikasinya sendiri serta menambahkan modul yang diperlukan sesuai kebutuhan. Flask

memiliki berbagai ekstensi yang dikembangkan oleh komunitas sehingga dapat menambahkan berbagai fungsi dengan mudah[4].

Flask memiliki kelebihan yaitu sangat ringan dan sangat sederhana dalam proses pengembangan. Sehingga Flask sangat cocok digunakan untuk pembuatan *HTTP Rest API*. Pada tugas akhir ini, Flask akan digunakan untuk pembuatan *HTTP Rest API*.

Pada tugas akhir ini FLask akan digunakan untuk mengatur end-point yang digunakan sistem untuk menerjemahkan perintah dari pengguna sistem.

2.3 Gitpython

Gitpython adalah pustaka python untuk berinteraksi dengan repositori git, dalam interaksi level tinggi seperi git-porcelain maupun level rendah seperti git-plumbing.

Gitpython menyediakan konsep dari obyek git untuk mempermudah mengakses data repositori dan juga mampu untuk mengakses repositori git secara langsung baik dengan implementasi python atau dengan menggunkan commnand git.[5]

Pada tugas akhir ini gitpython akan digunakan melakukan eksekusi perintah git yang akan digunakan oleh sistem dalam mengatur pelacakan konfigurasi perangkat jaringan.

2.4 Python Watchdog

Python watchdog adalah modul dari python untuk mengawasi *event* dari suatu file system .Watchdog dapat menangkap semua operasi yang terjadi di dalam direktori yang di tentukan.Watchdog bekerja secara real time dengan menggunakan thread.[6]

Pada tugas akhir ini python watchdog akan digunakan untuk melihat perubahan yang terjadi pada file konfigurasi perangkat jaringan yang disimpan oleh sistem.

2.5 TFTP

Trivial File Transfer Protocol (TFTP) adalah protokol pengiriman file yang sederhana tanpa menggunakan autentikasi pengguna. TFTP menggunakan protokol UDP dalam pengiriman data. TFTP tidak bisa digunakan untuk melihat isi dari suatu direktori, TFTP hanya bisa digunakan untuk mengirim dan menerima data.[7]

Dalam tugas akhir ini TFTP akan digunakan untuk jalur pengiriman file konfigurasi perangkat jaringan yang menggunakan protokol TFTP.

2.6 FTP

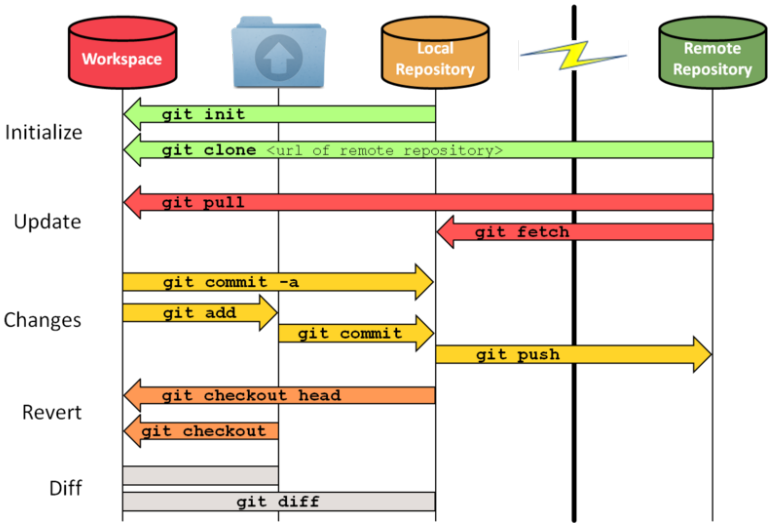
File Transfer Protocol (FTP) adalah protokol untuk mengirim file dari *server* ke *client* dengan menggunakan model *client-server*. FTP menggunakan protokol TCP dalam pengiriman *file*. FTP memiliki dua mode yaitu mode pasif dan mode aktif yang menentukan bagaiman client dan server terhubung.[8]

Dalam tugas akhir ini FTP akan digunakan untuk menerima konfigurasi yang dikirim oleh perangkat jaringan melalui protokol FTP.

2.7 Git

Git merupakan salah satu jenis dari VCS (*Version Control System*) yang banyak digunakan oleh pengembang aplikasi saat

ini. Git merupakan VCS yang gratis dan sumber terbuka. Git didesain untuk menangani proyek dari yang sederhana hingga yang kompleks dengan kecepatan dan efisiensi yang baik[9]. Git memiliki dua jenis *repository* yakni *local repository* dan *remote repository*. *Local repository* merupakan *repository* yang berada pada komputer kita dan *remote repository* adalah *repository* yang berada di server lain yang juga terhubung dengan *local repository* sehingga data yang ada pasti sama.



Gambar 2.1: Alur kerja git [10]

2.8 Gitea

Gitea adalah sebuah self-hosted layanan git atau layanan git yang bisa diimplementasikan sendiri sesuai kebutuhan. gitea memiliki antar muka dan fitur yang serupa dengan layanan git seperti bitbucket dan github. Gitea dibuat untuk menyediakan layanan git secara mudah dan cepat. Gitea menggunakan bahasa

Go dengan binary yang independen sehingga dapat bejalan di banyak platform yang mendukung bahasa Go.

(Halaman ini sengaja dikosongkan)

BAB III

DESAIN DAN PERANCANGAN

Pada bab ini dibahas mengenai analisis dan perancangan sistem.

3.1 Deskrisi Umum Sistem

Sistem yang akan dibuat dalam tugas akhir ini adalah sistem yang digunakan untuk melacak perubahan konfigurasi perangkat jaringan. Sistem terhubung dengan perangkat jaringan dan menyimpan semua versi perubahan dari file konfigurasi perangkat jaringan. Sistem bisa mengatur versi konfigurasi yang dibutuhkan oleh perangkat jaringan untuk dipasang pada perangkat jaringan.

Sistem memiliki *repository adapter* yang berfungsi untuk menerima file konfigurasi yang dikirim dari perangkat jaringan yang terhubung. Perangkat jaringan mengirimkan file konfigurasi menggunakan protokol pengiriman yang didukung seperti FTP, TFTP, dan SCP. Untuk menyimpan perubahan dari konfigurasi, sistem memiliki repositori git yang berfungsi menyimpan setiap catatan perubahan yang terjadi pada konfigurasi. Setiap perangkat yang terhubung akan disediakan repositori git untuk masing-masing perangkat. Sistem juga memiliki *repository observer* yang berfungsi untuk melihat setiap perubahan yang terjadi pada konfigurasi perangkat jaringan, setiap ada perubahan maka sistem akan otomatis mencatat perubahan ke dalam repositori git.

Di dalam sistem terdapat dua manajemen konsol yang digunakan yaitu Gitea dan Flask. Manajemen konsol tersebut digunakan untuk menerjemahkan instruksi dari administrator kepada sistem sesuai dengan diagram penggunaan pada Gambar 3.1.

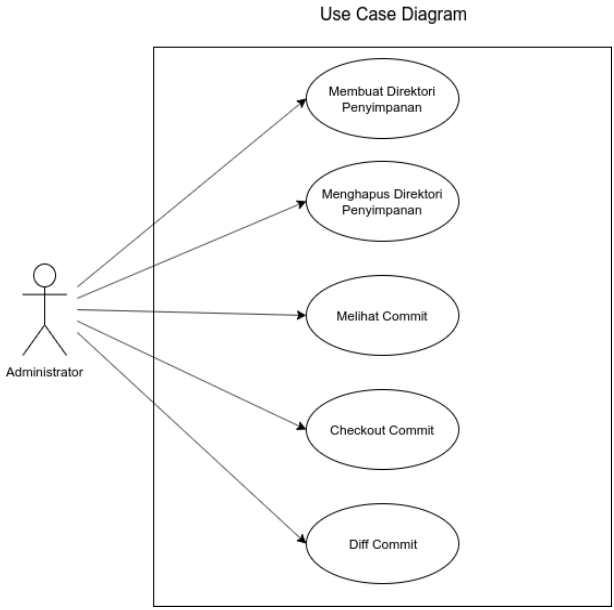
Sistem menggunakan basisdata untuk menyimpan data perangkat yang terhubung. Basis data yang digunakan adalah

basisdata mysql.

Proses penyimpanan konfigurasi dimulai dengan administrator mengirim konfigurasi dari perangkat jaringan menuju sistem. Sistem menerima *file* konfigurasi kemudian mencatat perubahan ke dalam *git repository*. Perubahan disimpan pada *local repository* dan *remote repostory*.

3.2 Kasus Penggunaan

Dalam sistem ini hanya ada satu aktor yaitu *administrator* jaringan yang akan mengatur penyimpanan konfigurasi. Terdapat lima fitur utama yang bisa digunakan dalam sistem. Diagram kasus penggunaan sistem pelacakan konfigurasi dapat dilihat pada Gambar 3.1.



Gambar 3.1: Diagram kasus penggunaan

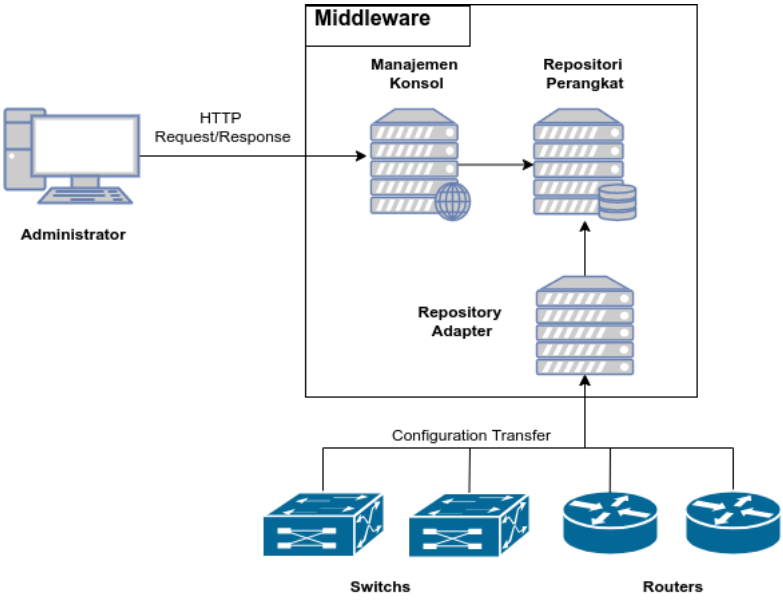
Diagram kasus penggunaan pada Gambar 3.1 dideskripsikan masing-masing pada Tabel 3.1.

Tabel 3.1: Daftar kode kasus penggunaan

Kode Kasus Penggunaan	**Nama Kasus Penggunaan**	**Keterangan**
UC-0001	Membuat Direktori Penyimpanan.	*Administrator* dapat membuat direktori untuk menyimpan konfigurasi dari perangkat jaringan.
UC-0002	Menghapus Direktori Penyimpanan.	*Administrator* dapat menghapus direktori penyimpanan jika sudah tidak digunakan.
UC-0003	Melihat Commit.	*Administrator* dapat melihat riwayat commit dalam repositori perangkat.
UC-0004	Checkout Commit.	Administrator dapat berpindah commit (checkout) sesuai dengan versi commit yang diinginkan.
UC-0005	Diff Commit.	Administrator dapat melihat perbedaan antara commit satu dengan lainnya.

3.3 Arsitektur Sistem

Pada sub-bab ini, dibahas mengenai tahap analisis dan kebutuhan bisnis serta desain dari sistem yang akan dibangun. Arsitektur sistem secara umum ditunjukkan pada Gambar 3.2.



Gambar 3.2: Desain umum sistem

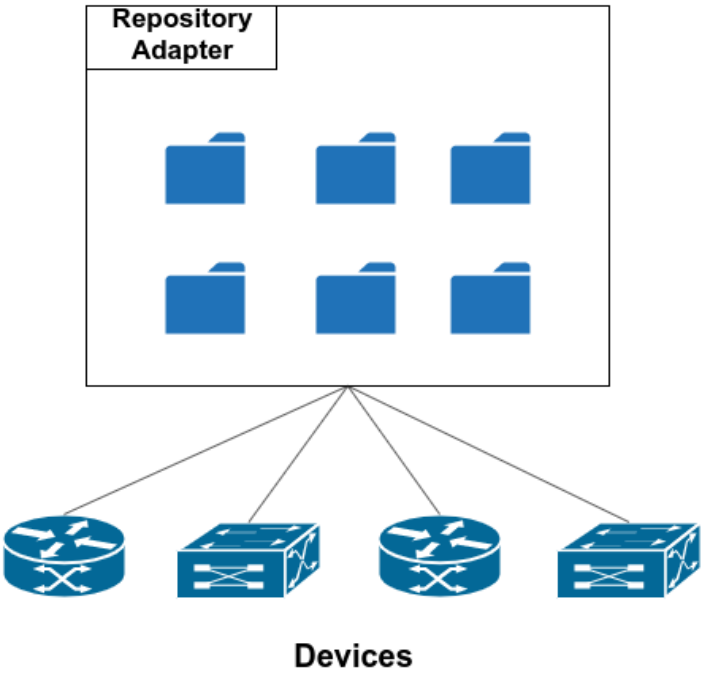
3.3.1 Desain Umum Sistem

Berdasarkan yang dijelaskan pada deskripsi umum sistem, dapat diperoleh kebutuhan sistem sebagai berikut:

1. *Repository Adapter* untuk menerima konfigurasi yang dikirim dari perangkat jaringan.
2. Repositori Perangkat untuk menyimpan file konfigurasi dari perangkat jaringan.

3. *Repository Observer* untuk melihat perubahan file yang disimpan di dalam repositori perangkat.
4. Manajemen Konsol untuk menerjemahkan intruksi dari admin kepada sistem.

3.3.2 Perancangan *Repository Adapter*



Gambar 3.3: Desain repositori adapter

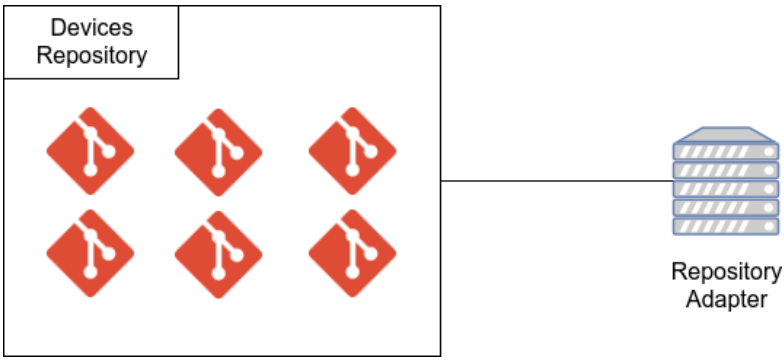
Repository adapter adalah komponen untuk menerima file konfigurasi yang dikirim dari perangkat jaringan menuju sistem. Sistem dapat menerima pengiriman konfigurasi melalui protokol pengiriman TFTP, FTP, dan SCP. Di dalam *repository adapter* setiap perangkat jaringan yang terhubung memiliki direktori

masing-masing. Setiap direktori dikondisikan mampu untuk menerima pengiriman konfigurasi dari *multi-protocol* sehingga perangkat jaringan tidak terbatas satu protokol saja dalam mengirim konfigurasi.

Di dalam *repository adapter* terdapat komponen *repository observer* yang berfungsi untum melihat file yang dikirim dari perangkat jaringan. Ketika ada file konfigurasi masuk maka *repository observer* akan otomatis memindahkan file konfigurasi ke dalam repositori perangkat.

Setelah file konfigurasi dipindahkan perubahan file konfigurasi dicatat oleh komponen repositori perangkat.

3.3.3 Perancangan Repositori Perangkat



Gambar 3.4: Desain repositori perangkat

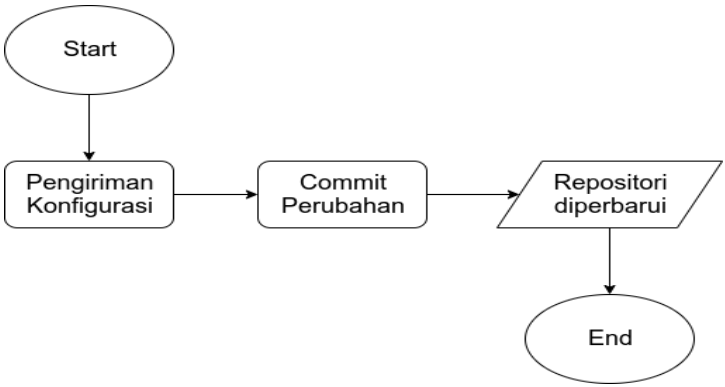
Repositori perangkat merupakan komponen untuk mencatat seluruh perubahan yang dilakukan pada file konfigurasi. Repositori perangkat berupa direktori yang berperan sebagai repoitori git. Setiap ada konfigurasi yang dikirim dari perangkat jaringan, konfigurasi akan diterima oleh *repository adapter* kemudian dipindahkan ke dalam repositori perangkat untuk dicatat perubahannya menggunakan git. Di dalam repositori

perangkat terdapat dua macam repositori git yakni *local repository* dan *remote repository*. Local repository terhubung dengan manajemen konsol yang berbasis Flask. *Remote repository* terhubung dengan manajemen konsol berbasis gitea.

Di dalam repositori perangkat terdapat terdapat komponen *repository observer* yang melihat perubahan file konfigurasi. Setiap ada file konfigurasi yang dipindahkan dari *repository adapter* maka akan diidentfikasi sebagai perubahan konfigurasi di dalam repositori perangkat.

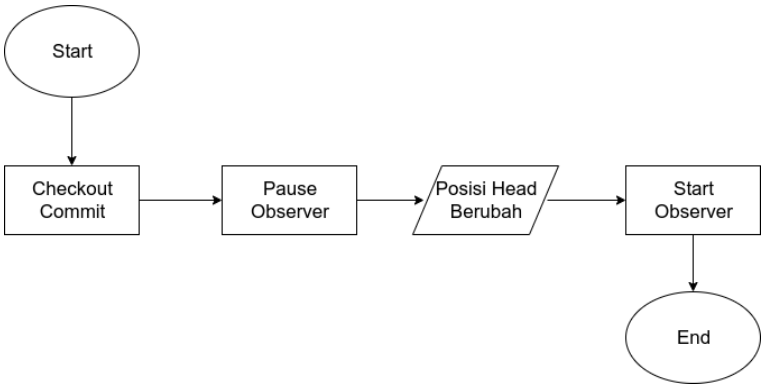
3.3.4 Perancangan *Repository Observer*

Pada sistem ini *Middleware* harus bisa mengamati repositori perangkat jaringan secara berkelanjutan dan melakukan update pada *history* commit pada repositori. Untuk melakukan hal tersebut modul watchdog di dalam *middleware* yang akan melihat setiap perubahan pada respositori perangkat jaringan. Modul watchdog berjalan sebagai thread yang menunggu perubahan kondisi di dalam repositori. Ketika thread mengidentifikasi ada perubahan di dalam repositori maka thread akan menjalankan perintah commit menggunakan modul gitPython yang terintegrasi dengan middleware. Alur dari repository observer dalam mencatat perubahan dapat dilihat pada gambar3.5.



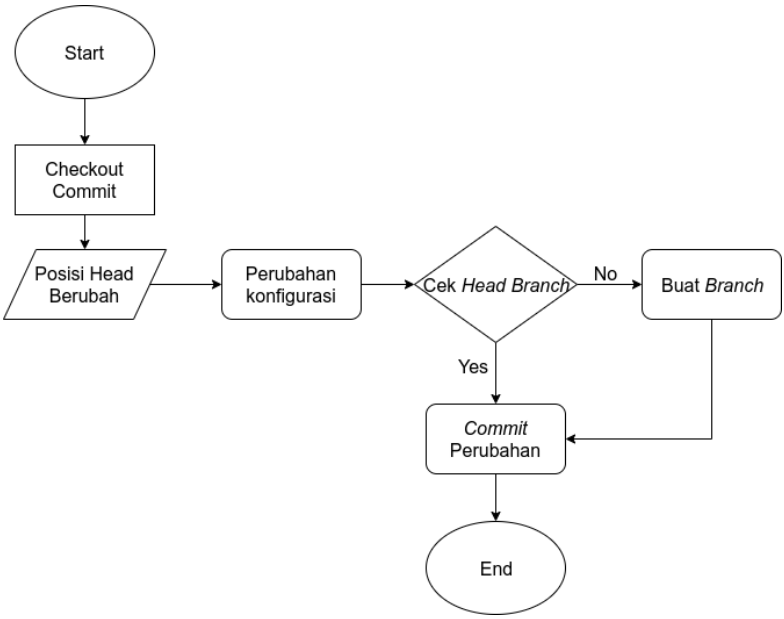
Gambar 3.5: Alur pengiriman file

Salah satu fungsi yang ada dalam sistem adalah fungsi *checkout commit* atau pindah versi konfigurasi. Untuk menjalankan fungsi tersebut *repository observer* harus berhenti sementara waktu hinnga posisi head berubah. Setelah posisi head berubah repository observer dijalankan kembali. Hal ini dilakukan agar proses perpindahan versi tidak dianggap perubahan baru yang menyebabkan ada commit baru pada repository. Alur proses checkout dapat dilihat pada gambar 3.6.



Gambar 3.6: Alur pemindahan versi

Repository Observer juga mengatur pembentukan cabang dari repositori penyimpanan konfigurasi perangkat jaringan. Pembuatan cabang dalam *history* commit diperlukan ketika posisi versi bukan merupakan versi dari perubahan terakhir dalam repositori. Alur pembuatan cabang dari repositori seperti pada gambar 3.7.



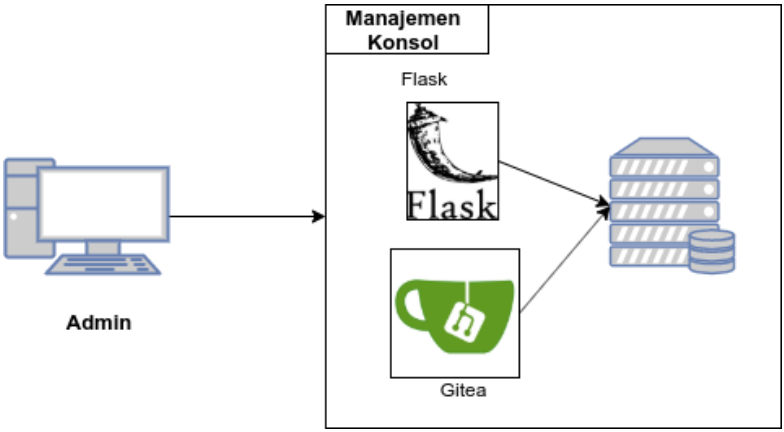
Gambar 3.7: Alur pembuatan branch

3.3.5 Perancangan Manajemen Konsol

Dalam sistem yang dibangun, manajemen konsol digunakan untuk menerjemahkan permintaan dari adiministrator jaringan. Manajemen konsol memiliki antarmuka dan rute dengan parameter nama repositori dan permintaan fitur yang diinginkan. Setiap rute akan diproses oleh *Middleware* dan kemudian mengirimkan respon kepada administrator.

Terdapat dua manajemen konsol yang digunakan dalam sistem pelacakan konfigurasi perangkat jaringan yakni Gitea *webapp* dan Flask. Flask digunakan untuk menampilkan *user interface* dari sistem, akan tetapi flask memiliki keterbatasan dalam fitur melihat isi file konfigurasi dan melihat diff commit atau perbedaan antara commit. Untuk mengatasi permasalahan

tersebut maka digunakan Gitea webapp untuk melihat isi file dan perbedaan antara commit.



Gambar 3.8: Perancangan manajemen konsol

Desain tampilan web untuk manajemen Konsol dapat dilihat pada gambar3.9.

Gambar 3.9: Perancangan tampilan manajemen konsol

3.3.6 Perancangan Basis Data

Dalam sistem diperlukan basis data untuk menyimpan data-data yang diperlukan. Basis data digunakan untuk menyimpan data dari perangkat jaringan yang terhubung dengan sistem. Oleh karena itu maka dibutuhkan tabel *device* untuk menyimpan perangkat yang terhubung serta versi konfigurasi yang disimpan. Di dalam basis data data yang disimpan adalah nama perangkat yang terhubung, alamat ip perangkat, serta versi terakhir dari konfigurasi perangkat.

BAB IV

IMPLEMENTASI

Pada bab ini akan dibahas implementasi dari perancangan setiap komponen sistem pada bab sebelumnya. Setiap komponen akan dibahas proses pembuatan dilengkapi dengan konfigurasi dan pseudocode dari sistem.

4.1 Lingkungan Implementasi

Dalam mengimplementasikan sistem, digunakan beberapa perangkat pendukung sebagai berikut.

4.1.1 Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem adalah sebagai berikut:

1. Virtual Private Server dengan CPU Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz dan RAM 8GB.

4.1.2 Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan adalah sebagai berikut:

1. Sistem Operasi Ubuntu 18.04 LTS 64 Bit.
2. Flask versi 1.0.3 untuk pengembangan manajemen konsol.
3. Git versi 2.17.1 untuk *versioning* file konfigurasi.
4. TFTP untuk protokol pengiriman file konfigurasi.
5. FTP untuk protokol pengiriman file konfigurasi.
6. Gitea untuk pengembangan manajemen konsol.

4.2 Implementasi *Repository Adapter*

Repositori perangkat mendukung tiga protokol pengiriman file yaitu File Transfer Protocol (FTP), Trivial File Transfer Protocol (TFTP) dan Secure Copy Protocol SCP (SCP). Ada

beberapa tahap agar protokol-protokol tersebut bisa digunakan yakni pemasangan dan konfigurasi. Untuk implementasi Reposi Perangkat terdapat dua tahap yakni :

1. Implementasi direktori untuk protokol TFTP.
2. Implementasi direktori untuk protokol FTP.

4.2.1 Implementasi Protokol TFTP

Untuk melakukan pemasangan TFTP bisa dilihat di lampiran kode sumber . Setelah selesai melakukan pemasangan maka kita perlu melakukan konfigurasi TFTP pada file `/etc/default/tftpd-hpa`.

```
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/home/didin/REPO/"
TFTP_ADDRESS=":69"
TFTP_OPTIONS="--secure --create"
```

Kode Sumber IV.1: Konfigurasi TFTP

Dari pengaturan diatas menunjukkan bahwa protokol TFTP menggunakan username ’tftp’. Direktori untuk menyimpan file yang dikirim dari perangkat jaringan disimpan pada path `/home/didin/REPO/`. Port yang digunakan untuk koneksi TFTP adalah port 69. Parameter options ’–secure’ digunakan untuk mengisolasi directory yang bisa diakses menggunakan tftp sehingga file system yang lain tidak terganggu. Parameter ’–create’ digunakan agar meskipun belum ada file di dalam direktori server tftp client atau perangkat jaringan tetap bisa mengirim file konfigurasi.

Setelah melakukan konfigurasi TFTP selanjutnya adalah melakukan konfigurasi pada direktori yang digunakan sebagai *root* TFTP dengan menjalankan perintah berikut.

```
chown -R didin:tftp /home/didin/REPO
```

Kode Sumber IV.2: Konfigurasi direktori TFTP

Perintah diatas berfungsi agar direktori `/home/didin/REPO` bisa digunakan untuk mengirim konfigurasi dengan user 'didin' atau group 'tftp'.

4.2.2 Implementasi Protokol FTP

Selanjutnya dalam implementasi Repositori perangkat adalah pemasangan dan konfigurasi FTP. Untuk melakukan pemasangan FTP jalankan dapat dilihat di lampiran kode sumber. Setelah melakukan pemasangan langkah selanjutnya adalah mengaktifkan port yang digunakan dalam FTP yakni port 20 dan 21 dengan menjalankan perintah.

```
sudo ufw allow 20/tcp
sudo ufw allow 21/tcp
sudo ufw status
```

Kode Sumber IV.3: Aktivasi port FTP

Tahap selanjutnya adalah mengatur konfigurasi dari FTP pada file `/etc/vsftpd.conf` dengan menulis konfigurasi berikut.

```
anonymous_enable=NO
local_enable=YES
write_enable=YES
local_umask=022
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
xferlog_std_format=YES
listen=NO
listen_ipv6=YES
pam_service_name=vsftpd
userlist_enable=YES
tcp_wrappers=YES
```

```
userlist_enable=YES
userlist_file=/etc/vsftpd.userlist
userlist_deny=NO
chroot_local_user=YES
allow_writeable_chroot=YES
local_root=/home/$USER/REPO
```

Kode Sumber IV.4: Konfigurasi file FTP

Konfigurasi diatas berfungsi untuk :

1. Direkttori `/home/USER/REPO` untuk menyimpan konfigurasi yang dikirim dari perangkat jaringan.
2. Mengatur *permission* direktori agar bisa digunakan untuk menyimpan.
3. Mengatur agar *user* tertentu saja yang bisa mengirim file melalui FTP.
4. Mengisolasi agar koneksi FTP hanya berjalan di direktori untuk menyimpan file saja dan tidak memiliki akses ke *filesystem* utama.

User yang memiliki otoritas untuk melakukan koneksi FTP harus didefinisikan di dalam file vsftpd.userlist. Oleh karena itu tambahkan nama pengguna yang punya otoritas untuk FTP di dalam file `/etc/vsftpd.userlist` dengan menjalankan perintah.

```
echo "didin" | sudo tee -a /etc/vsftpd.
   userlist
```

Kode Sumber IV.5: Pengguna FTP

Untuk menerapkan konfigurasi jalankan ulang FTP dengan menjalankan perintah.

```
systemctl restart vsftpd
```

Kode Sumber IV.6: Jalan ulang FTP

4.3 Implementasi Repositori Perangkat

Komponen repositori perangkat digunakan untuk menyimpan catatan perubahan file konfigurasi perangkat jaringan. Untuk menyiapkan komponen repositori perangkat pertama-tama install git. Instalasi git dapat dilihat pada lampiran A. Komponen repossitori perangkat berjalan pada direktori `/home/didin/REPO/downloads`. Direktori tersebut dikondisikan untuk bisa diakses menggunakan TFTP, FTP, dan SCP karena dibutuhkan agar perangkat jaringan bisa mengunduh file konfigurasi dari sistem.

4.4 Implementasi *Repository Observer*

Middleware memiliki tugas untuk mencatat setiap perubahan yang terjadi pada file konfigurasi. Perubahan file konfigurasi terjadi ketika perangkat jaringan mengirim file konfigurasi menuju *middleware*. Untuk mengamati perubahan dalam direktori terdapat *Repository Observer* dalam bentuk thread. Untuk membuat *repository observer* ada beberapa tahap yang diperlukan yakni pemasangan bahasa python dan modul-modul yang diperlukan kemudian pembuatan program untuk menjalankan *thread repository observer*. Perangkat lunak yang diperlukan untuk dipasang adalah:

1. Python.
2. Python Watchdog.
3. Git Python.

Berikut pseudocode yang berjalan dalam *Repository Observer*.

Pseudocode 1: Repository observer

```
if file modified then
    if head not a branch head then
        create new branch;
    else
        reference head to branch;
    end
    git add;
    git commit;
    git push;
end
if checkout then
    pause observer 2 second
end
```

4.5 Implementasi Manajemen Konsol

Manajemen konsol pada *middleware* berfungsi untuk menjembatani antara pengguna dengan *middleware*. Pengguna mengirimkan permintaan melalui rute-rute yang dimiliki manajemen konsol kemudian permintaan diproses oleh *midleware*. Dalam sistem yang dibuat ini web service yang digunakan adalah Gitea yang merupakan self-hosted git dan juga Flask.

4.5.1 Implementasi Flask

Untuk menggunakan Flask ada beberapa tahap yang harus dilakukan terlebih dahulu yakni.

1. Instalasi bahasa Python.
2. Instalasi modul Flask.
3. Instalasi modul Flask-SQLAlchemy.

Berikut rute yang disediakan oleh Flask 4.1.

Tabel 4.1: Tabel rute manajemen konsol flask

No	Rute	*Method*	Keterangan
1	`/register`	Post	Membuat user baru.
2	`/login`	Post	Login ke halaman dashboard sistem.
3	`/home`	Get	Menampilkan perangkat yang terhubung dengan sistem.
4	`/home`	Post	Membuat repositori untuk perangkat jaringan.
5	`/{repo}/ branch/ {branchname}`	Get	Menampilkan daftar commit dari repositori.
6	`/delete/ {reponame}`	Get	Menghapus repositori.

4.5.2 Implementasi Gitea

Untuk menggunakan Gitea ada beberapa tahap yang harus dilakukan terlebih dahulu yakni :

1. Instalasi bahasa Go.
2. Instalasi database Mysql.
3. Pembuatan database gitea.
4. Clone kode sumber Gitea.

Instalasi bahasa Go dapat dilihat di A.1.

Untuk menggunakan fitur-fitur dalam sistem yang dibuat pengguna perlu mengakses rute-rute dalam *Web Service*. Berikut rute yang disediakan Gitea webapp pada Tabel 4.2.

Tabel 4.2: Tabel rute manajemen konsol gitea

No	Rute	*Method*	Keterangan
1	`/user/sign_up`	Post	Membuat user untuk administrator.
2	`/user/login`	Post	Login user administrator.
3	`/repo/create`	Post	Membuat repositori Gitea.
4	`/{username}/{reponame}`	Get	Menampilkan repositori dari user.
5	`/{username}/{reponame}/commits/branch/{namabranch}`	Get	Menampilkan commit pada repositori.

4.5.3 Implementasi Skema Basis Data

Berdasarkan perancangan sistem pada bab sebelumnya data akan disimpan pada basis data MySQL. Data yang akan disimpan adalah data dari perangkat yang terhubung dengan sistem. Rincian tabel perangkat dapat dilihat pada tabel.

Tabel 4.3: Tabel device

No	Kolom	Tipe	Keterangan
1	id	int	sebagai *primary key*.
2	device_name	varchar(100)	Menunjukkan nama perangkat.
3	device_ip	varchar(100)	Menunjukkan alamat IP dari perangkat.
4	device_repo	varchar(100)	Menunjukkan path *repositori* perangkat.

Tabel 4.3: Tabel device

No	Kolom	Tipe	Keterangan
5	device_version	varchar(100)	Menunjukkan versi terkini dari konfigurasi yang disimpan.

(Halaman ini sengaja dikosongkan)

BAB V

PENGUJIAN DAN EVALUASI

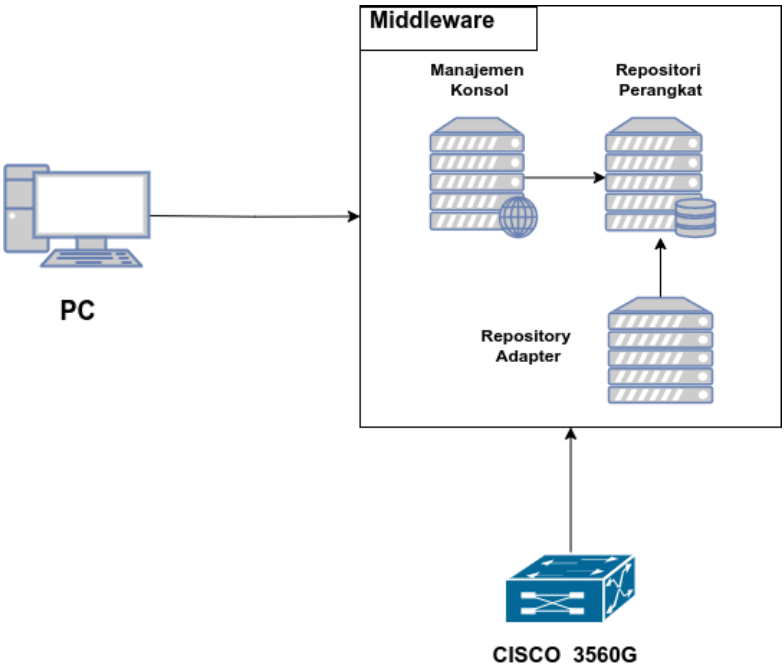
Pada bab ini akan dibahas uji coba dan evaluasi dari sistem yang sudah dibuat. Sistem akan diuji coba fungsionalitasnya dengan menjalankan skenario pengujian fitur-fitur dari sistem yang dibuat. Sistem juga akan diuji coba performa dengan skenario pengujian beban terhadap sistem. Uji coba dilakukan untuk mengevaluasi kinerja dari sistem dengan lingkungan uji coba yang ditentukan.

5.1 Lingkungan Uji Coba

Lingkungan uji coba sistem ini terdiri dari beberapa komponen yaitu *middleware* dan perangkat jaringan yang terhubung. Server yang digunakan sebagai *middleware* merupakan *Virtual Private Server* yang disediakan oleh DPTSI ITS. Perangkat jaringan yang digunakan adalah Cisco Catalyst 3560G. Spesifikasi dari *Middleware* bisa dilihat di tabel 5.1

Tabel 5.1: Spesifikasi Komponen

No	Komponen	Perangkat Keras	Perangkat Lunak
1	*Middleware*	4 Core Processor, 4GB RAM, HDD 64 GB	Ubuntu 18.04.3, MySQL 5.7, Python 3.6, Go1.13.5, Flask 1.0.3, Python 3.6
2	Switch	512KB RAM, 38.7 Mpps, 2 Virtual Ethernet interfaces, 28 Gigabit Ethernet interfaces	ios Version 12.2(53)SE2
3	PC	8 Core Processor, 8GB RAM, SSD 128	Ubuntu 18.04, Python 3.6



Gambar 5.1: Penggunaan storage repositori

5.2 Skenario Uji Coba

Uji coba ini dilakukan untuk menguji fungsionalitas dari sistem yang dibuat telah sesuai dengan perancangan dan sistem benar-benar diimplementasikan dan bekerja sesuai seharusnya. Skenario pengujian dibedakan menjadi 2 bagian yaitu:

- **Uji Fungsionalitas**
  Pengujian yang dilakukan berdasarkan fungsionalitas yang disediakan sistem.
- **Uji Performa**
  Pengujian yang dilakukan untuk melihat waktu yang diperlukan untuk menyimpan konfigurasi perangkat

jaringan.

5.2.1 Skenario Uji Coba Fungsionalitas

Uji fungsionalitas dibagi menjadi beberapa bagian antara lain yaitu *user* mengelola repositori, *user* mengirim file konfigurasi, *user* melakukan checkout commit, mengunduh file setelah checkout commit, dan percabangan commit.

5.2.1.1 Uji Fungsionalitas *User* Mengelola Repositori

Uji coba ini bertujuan untuk memastikan fitur dari mengelola repositori dapat dijalankan dengan benar. Uji coba dilakukan dengan user mengakses sistem melalui rute untuk mengelola repositori. *User* mengirimkan request ke manajemen konsol. Rancangan pengujian dan hasil yang diinginkan dapat dilihat pada Tabel .

Tabel 5.2: skenario uji fungsionalitas user mengelola repositori

No	Rute	Uji Coba	Harapan
1	`/home`	Mengirimkan request Get menuju manajemen konsol	Request berhasil diterima manajemen konsol, kemudian manajemen konsol meampilkan halaman daftar perangkat yang dihubungkan dan menampilkan form untuk menambahkan perangkat.

No	Rute	Uji Coba	Harapan
2	`/home`	Mengirimkan request Post menuju manajemen konsol	Request berhasil diterima manajemen konsol, kemudian manajemen konsol membuat repositori untuk perangkat yang ditambahkan.
3	`/{reponame}/branch/{branchname}`	Mengirimkan request menuju manajemen konsol	Request berhasil diterima manajemen konsol, kemudian manajemen konsol menampilkan daftar commit pada repositori.
4	`/{username}/{reponame}/commit/{hashcommit}`	Mengirimkan request menuju manajemen konsol	Request berhasil diterima manajemen konsol, kemudian manajemen konsol menampilkan diff commit dari hash yang dipilih dengan *commit parent*-nya.
5	`/delete/{reponame}`	Mengirimkan request menuju manajemen konsol	Request berhasil diterima manajemen konsol, kemudian sistem menghapus repositori.

5.2.1.2 Uji Fungsionalitas User Mengirim Konfigurasi

Uji coba ini bertujuan untuk memastikan perangkat jaringan dapat mengirim file konfigurasi ke dalam repositori sistem. Juga

untuk memastikan setiap ada perubahan pada konfigurasi perangkat maka sistem otomatis melakukan commit pada git repositori.

Uji coba ini dilakukan dengan cara user mengirimkan file konfigurasi dari perangkat jaringan menuju *middleware* sistem. Pengiriman konfigurasi menggunakan protokol yang didukung oleh perangkat jaringan. Setelah file dikirim, *middleware* akan melihat ada perubahan dalam repositori sehingga *middleware* langsung menjalankan perintah commit dan push. Setelah commit dilakukan dapat dilihat histori commit dari repositori.

Uji coba berhasil ketika file konfigurasi berhasil terkirim dan pada repositori terbuat commit baru.

5.2.1.3 Uji Fungsionalitas *User* Merubah Versi

Uji coba ini bertujuan untuk memastikan admin dapat merubah versi commit dari file konfigurasi di dalam repositori.

Uji coba ini dilakukan dengan cara user me-klik tombol pilih versi pada daftar commit di dalam repositori. Sistem akan melakukan checkout commit pada repositori lokal sehingga versi konfigurasi berubah sesuai versi yang dipilih. Uji coba perubahan versi dilakukan pada *branch* yang sama dan *branch* yang berbeda.

Uji coba berhasil ketika versi dari file konfigurasi berhasil berubah sesuai dengan yang diinginkan.

5.2.1.4 Uji Fungsionalitas Unduh Konfigurasi

Uji coba ini bertujuan untuk memastikan perangkat jaringan dapat mengunduh file konfigurasi dari *middleware* sistem. Uji coba juga untuk memastikan perangkat jaringan bisa mengunduh semua versi yang ada di *middleware* sistem.

Uji coba ini dilakukan dengan cara user mengakses perangkat jaringan yang terhubung dengan *Middleware*. User

kemudian mengunduh file konfigurasi dari *middleware* kedalam perangkat jaringan. File konfigurasi diunduh sebelum versi dirubah dan setelah versi dirubah.

Uji coba berhasil ketika perangkat jaringan bisa mengunduh file konfigurasi dari *middleware* sistem.

5.2.1.5 Uji Fungsionalitas Percabangan Commit

Uji coba ini bertujuan untuk memastikan ketika perangkat jaringan mengirim konfigurasi dan kondisi versi bukan merupakan versi terbaru maka akan terbentuk cabang baru pada commit repositori.

Uji coba ini dilakukan dengan cara user melakukan checkout pada repositori. Setelah checkout, user kemudian mengakses perangkat jaringan dan mengirimkan file konfigurasi ke *middleware*. Karena posisi head tidak berada pada posisi commit terbaru maka sistem akan otomatis membuat cabang baru setelah ada perubahan di dalam repositori.

Uji coba berhasil ketika setelah pengiriman file konfigurasi, repositori perangkat otomatis membuat cabang baru.

5.2.2 Skenario Uji Coba Performa

Uji performa digunakan untuk menguji bagaimana ketahanan sistem dalam menyimpan konfigurasi perangkat jaringan dan mengatur versi perangkat jaringan.

Uji performa dilakukan dengan cara mengirim konfigurasi dari perangkat jaringan menuju middlware sistem secara bertahap. Perubahan yang dilakukan pada konfigurasi dilakukakan secara bertahap mulai dari 50 perubahan hingga 500 perubahan dengan penambahan 50 perubahan di setiap tahap. Uji coba dilakukan untuk melihat beberapa storage yang digunakan oleh repositori untuk penyimpanan konfigurasi dan dalam menyimpan perubahan. Uji coba juga dilakukan untuk melihat

waktu yang diperlukan untuk merubah versi konfigurasi.

Hasil yang diharapkan dari pengujian ini adalah sistem memiliki *storage* yang cukup untuk menyimpan konfigurasi minimal selama satu tahun.

5.3 Hasil Uji Coba dan Evaluasi

Berikut ini dijelaskan hasil coba dan evaluasi berdasarkan skenario yang sudah dijelaskan pada subbab sebelumnya.

5.3.1 Uji Fungsionalitas

Berikut ini dijelaskan hasil dari pengujian fungsionalitas pada sistem yang sudah dibangun.

5.3.1.1 Uji Fungsionalitas User Mengelola Repositori

Uji coba dilakukan dengan mengakses manajemen konsol melalui rute yang ditentukan pada tabel 5.4. Hasil pengujian dapat dilihat pada tabel .

Tabel 5.3: Hasil uji fungsionalitas user mengelola repositori

No	Rute	Harapan	Hasil
1	`/home`	Request berhasil diterima manajemen konsol, kemudian manajemen konsol meampilkan halaman daftar perangkat yang dihubungkan dan menampilkan form untuk menambahkan perangkat.	OK

No	Rute	Harapan	Hasil
2	`/home`	Request berhasil diterima manajemen konsol, kemudian manajemen konsol membuat repositori untuk perangkat yang ditambahkan.	OK
3	`/{reponame}/branch/{branchname}`	Request berhasil diterima manajemen konsol, kemudian manajemen konsol menampilkan daftar commit pada repositori.	OK
4	`/{username}/{reponame}/commit/{hashcommit}`	Request berhasil diterima manajemen konsol, kemudian manajemen konsol menampilkan diff commit dari hash yang dipilih dengan *commit parent*-nya.	OK
5	`/delete/{reponame}`	Request berhasil diterima manajemen konsol, kemudian sistem menghapus repositori.	OK

5.3.1.2 Uji Fungsionalitas User Mengirim Konfigurasi

Uji coba ini dilakukan dengan cara user mengirim file konfigurasi dari perangkat jaringan menggunakan protokol TFTP, FTP, dan SCP. Keterangan hasil uji bisa dilihat pada tabel.

Tabel 5.4: Hasil uji fungsionalitas user mengelola repositori

No	Protokol	Harapan	Hasil
1	TFTP	File Konfigurasi terbuat di dalam repositori	OK
2	FTP	File Konfigurasi terbuat di dalam repositori	OK
3	SCP	File Konfigurasi terbuat di dalam repositori	OK

5.3.1.3 Uji Fungsionalitas *User* Merubah Versi

Uji coba ini dilakukan dengan cara user me-klik tombol pilih versi pada commit yang diinginkan. Keterangan hasil uji bisa dilihat pada tabel 5.5.

Tabel 5.5: Hasil uji fungsionalitas user checkout commit

No	Perubahan	Harapan	Hasil
1	Perubahan pada satu branch	Versi pada repositori berubah	OK
2	Perubahan pada branch berbeda	Versi pada repositori berubah	OK

5.3.1.4 Uji Fungsionalitas Unduh Konfigurasi

Uji coba dilakukan dengan cara administrator membuka perangkat jaringan dan mengunduh konfigurasi dari sistem. Keterangan hasil uji bisa dilihat pada tabel 5.6.

Tabel 5.6: Hasil uji fungsionalitas user checkout commit

No	Unduhan	Harapan	Hasil
1	Sebelum checkout	File konfigurasi berhasil diunduh	OK
2	Setelah checkout	File konfigurasi berhasil diunduh	OK

5.3.1.5 Uji Fungsionalitas Percabangan Commit

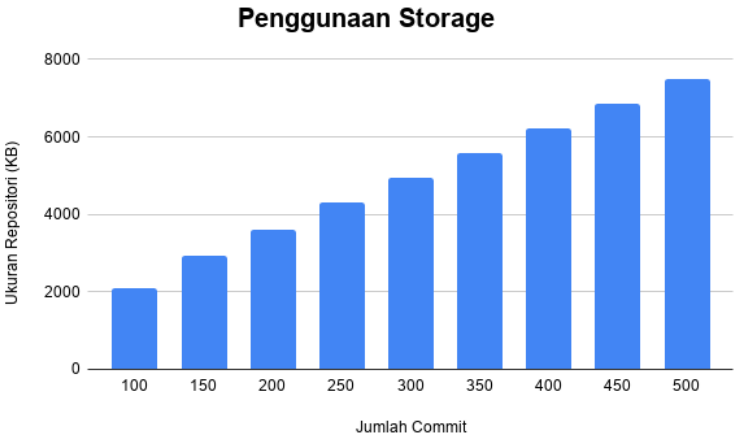
Uji coba ini dilakukan dengan cara mengunggah konfigurasi perangkat menuju sistem dengan kondisi versi dalam sistem bukan versi terbaru yang dicatat. Keterangan hasil ujicoba bisa dilihat pada tabel.

Tabel 5.7: Hasil uji fungsionalitas percabangan commit

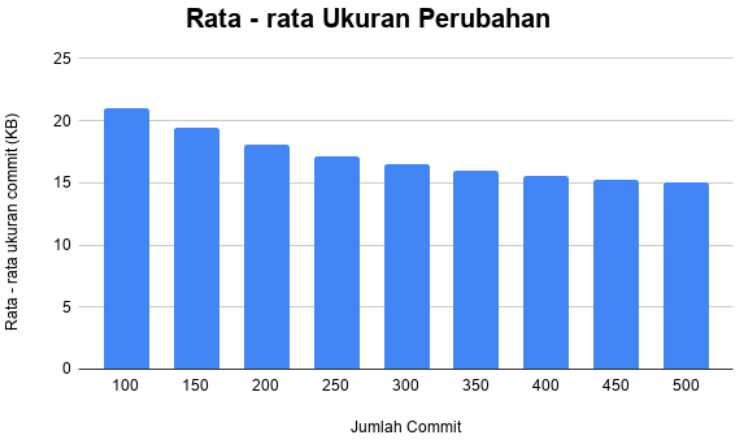
No	Urutan Cabang	Harapan	Hasil
1	1	Cabang terbuat	OK
2	2	Cabang terbuat	OK
3	3	Cabang terbuat	OK

5.3.2 Hasil Uji Performa

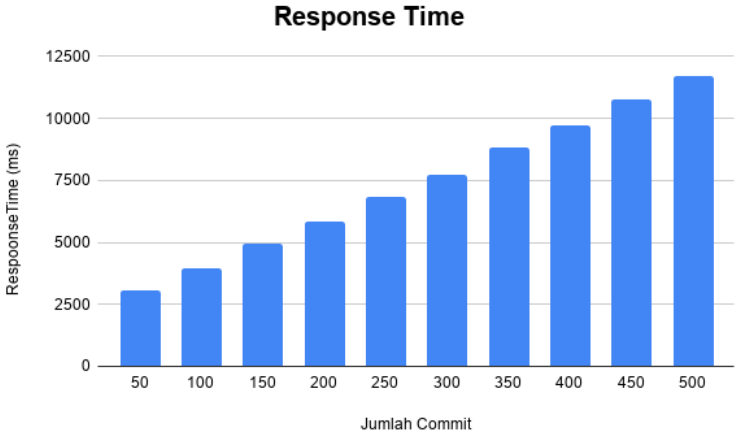
Uji performa dilakukan pada *middleware* sistem. Pengujian dilakukan dengan mengirimkan konfigurasi dari perangkat jaringan menuju *middleware*. Jumlah perubahan dimulai dari 50 perubahan sampe 500 perubahan dan ditambahkan secara bertahap dengan interval 50 perubahan. Hasil uji coba setelah dilakukan perubahan konfigurasi dari perangkat jaringan dan disimpan di *middleware* dapat dilihat pada Gambar5.2. Rata-rata storage yang digunakan untuk setiap perubahan dapat dilihat pada Gambar5.3. Dalam uji performa uga dilakukan pengujian untuk melihat response time ketika melakukan checkout (pemindahan versi) di dalam repositori. Hasil pengujian response time dapat dilihat pada grafik 5.4.



Gambar 5.2: Penggunaan storage repositori



Gambar 5.3: Penggunaan storage repositori



Gambar 5.4: Penggunaan storage repositori

Dari grafik 5.2 dapat dilihat bahwa penggunaan storage tidak mengalami perbedaan yang banyak. Dapat dilihat bahwa storage yang digunakan berubah linier terhadap jumlah commit (perubahan) yang dilakukan. Jika diasumsikan dalam setahun dilakukan perubahan setiap hari pada perangkat jaringan maka minimal storage yang dibutuhkan untuk mencatat perubahan satu perangkat adalah 6 MB.

Dari grafik 5.3 dapat dilihat bahwa rata-rata storage yang dibutuhkan untuk menyimpan perubahan semakin sedikit ketika jumlah perubahan semakin banyak. Dari uji coba yang dilakukan ukuran perubahan konfigurasi yang disimpan di dalam repositori adalah 15-25 KB.

Dari grafik 5.4 dapat dilihat bahwa jumlah perubahan yang yang dicatat mempengaruhi waktu yang diperlukan untuk melakukan checkout commit. Semakin banyak perubahan yang disimpan di dalam repositori maka waktu yang diperlukan untuk melakukan checkout (pemindahan versi) akan semakin lama. Hal

ini terjadi karena semakin banyak perubahan yang dicatat maka semakin banyak juga data perubahan yang diambil untuk ditampilkan.

(Halaman ini sengaja dikosongkan)

BAB VI

PENUTUP

Bab ini membahas kesimpulan yang dapat diambil dari tujuan pembuatan sistem dan hubungannya dengan hasil uji coba dan evaluasi yang telah dilakukan. Selain itu, terdapat beberapa saran yang bisa dijadikan acuan untuk melakukan pengembangan dan penelitian lebih lanjut.

6.1 Kesimpulan

Dari proses perencangan, implementasi dan pengujian terhadap sistem, dapat diambil beberapa kesimpulan berikut:

1. Sistem dapat menyimpan konfigurasi perangkat jaringan di dalam direktori yang dibedakan berdasarkan nama perangkat jaringan yang sudah ditentukan.
2. Sistem dapat menerima konfigurasi yang dikirim dari perangkat jaringan melalui protokol FTP, TFTP, dan SCP menyesuaikan protokol yang didukung perangkat jaringan.
3. Kebutuhan storage untuk menyimpan linier dengan jumlah perubahan yang dilakukan.
4. Sistem mengecek perubahan konfigurasi yang disimpan di dalam repositori menggunakan modul *python wathdog*.
5. Sistem dapat digunakan untuk merubah versi konfigurasi yang disimpan.
6. Konfigurasi yang disimpan dalam sistem dapat diunduh kembali oleh perangkat jaringan setelah versinya dirubah.
7. Sistem dapat digunakan untuk melihat perbedaan versi satu dengan yang lainnya.
8. Waktu yang dieperlukan untuk memindah versi bergantung kepada jumlah perubahan yang disimpan.

6.2 Saran

Berikut beberapa saran yang diberikan untuk pengembangan lebih lanjut:

1. Path atau url untuk mengunduh konfigurasi dijadikan sama dengan path untuk mengunggah konfigurasi.
2. Ditambahkan fitur untuk melihat struktur tree dari catatan commit sehingga memudahkan meliaht daftar perubahan ketika memiliki cabang lebih dari satu.
3. Melakukan optimasi waktu yang diperlukan untuk merubah versi konfigurasi yang disimpan.
4. Manajemen konsol yang menggunakan dua macam tampilan dijadikan satu macam tampilan.

DAFTAR PUSTAKA

[1] "Compare repositories," 5 Agustus 2019. [Daring]. Tersedia pada: https://www.openhub.net/repositories/compare. [Diakses: 5 Agustus 2019].

[2] "About - git," 5 Agustus 2019. [Daring]. Tersedia pada: https://git-scm.com/about/small-and-fast. [Diakses: 5 Agustus 2019].

[3] "General Python FAQ," 3 Mei 2018. [Daring]. Tersedia pada: https://docs.python.org/3/faq/general.html#what-is-python. [Diakses: 3 Mei 2018].

[4] "A simple framework for building complex web applications," 3 Mei 2018. [Daring]. Tersedia pada: https://pypi.org/project/Flask/. [Diakses: 3 Mei 2018].

[5] "GitPython overview," 3 Mei 2019. [Daring]. Tersedia pada: https://gitpython.readthedocs.io/en/stable/intro.html. [Diakses: 3 Mei 2019].

[6] "Watchdog," 29 Agustus 2019. [Daring]. Tersedia pada: https://pypi.org/project/watchdog/. [Diakses: 29 Agustus 2019].

[7] N. F. S. Pauzi, M. A. M. Isa, H. Hashim, S. F. S. Adnan, dan L. Mazalan, "Performance measurement of secure TFTP protocol for smart embedded devices," in *2014 IEEE Asia Pacific Conference on Wireless and Mobile*, Aug. 2014.

[8] N. Udayakumar, A. Khera, L. Suri, C. Gupta, dan T. Subbulakshmi, "Bandwidth analysis of file transfer protocol," in *2018 International Conference on Communication and Signal Processing (ICCSP)*, April 2018.

[9] "About - git," 22 Desember 2019. [Daring]. Tersedia pada: https://git-scm.com. [Diakses: 22 Desember 2019].

[10] “Subversion,” 23 Desember 2019. [Daring]. Tersedia pada: https://polyawesomism.wordpress.com/tag/subversion/. [Diakses: 23 Desember 2019].

LAMPIRAN A

INSTALASI PERANGKAT LUNAK

Instalasi Protokol Pengiriman File

- TFTP
  $ `sudo apt install tftp-hpa tftpd-hpa`
- FTP
  $ `sudo apt-get install vsftpd`

Instalasi Git

$ `sudo apt-get install Git`

Instalasi Bahasa Go

```
cd ~
curl -O https://dl.google.com/go/go1.13.5.linux
    -amd64.tar.gz

tar xvf go1.13.5.linux-amd64.tar.gz
sudo chown -R root:root ./go
sudo mv go /usr/local
```

Kode Sumber A.1: Instalasi Bahasa Go

Selanjutnya export path untuk bahasa Go di dalam `~/.profile`

```
export GOPATH=$HOME/go
export PATH=$PATH:/usr/local/go/bin:$GOPATH/bin
```

Kode Sumber A.2: Pengaturan Path

Instalasi Pustaka Python

Dalam pengembangan sistem ini, digunakan berbagai pustaka pendukung. Pustaka pendukung yang digunakan

merupakan pustaka untuk bahasa pemrograman Python. Berikut adalah daftar pustaka yang digunakan dan cara pemasangannya:

- Python
  $ `sudo apt-get install python3`
- Flask
  $ `sudo pip3 install Flask`
- Watchdog
  $ `sudo pip3 install watchdog`
- Gitpython
  $ `sudo pip3 install gitpython`

LAMPIRAN B

KODE SUMBER

Kode Sumber Pengamat Perubahan Repositori

Kode Sumber Observer.py

```
1  import threading
2  import sys
3  import time
4  import datetime
5
6  from watchdog.observers import Observer
7  from watchdog.events import
       FileSystemEventHandler
8  from watchdog.events import
       PatternMatchingEventHandler
9
10 from git import Repo, Git
11
12 from app.src.ApplicationRepo import
       ApplicationRepo as ar
13
14 class ObserverThread(threading.Thread):
15 def __init__(self,path,name):
16 threading.Thread.__init__(self)
17 self.repo_path = path
18 self.name = name
19
20 def pause_thread(self):
21 self.observer.pause()
22
23 def cont_thread(self):
24 self.observer.resume()
25
26
```

```
27    def run(self):
28    print("thread started")
29    self.event_handler = EventHandler(self.name,
          self.repo_path)
30
31    self.observer = PausingObserver()
32
33    self.observer.schedule(self.event_handler, self
          .repo_path, recursive=True)
34    self.observer.start()
35
36    try:
37    while True:
38    time.sleep(1)
39    except KeyboardInterrupt:
40    self.observer.stop()
41    self.observer.join()
42
43    class PausingObserver(Observer):
44    def dispatch_events(self, *args, **kwargs):
45    if not getattr(self, '_is_paused', False):
46    super(PausingObserver, self).dispatch_events(*
          args, **kwargs)
47
48    def pause(self):
49    self._is_paused = True
50
51    def resume(self):
52    time.sleep(2)  # allow interim events to be
          queued
53    self.event_queue.queue.clear()
54    self._is_paused = False
55
```

```
56  class EventHandler(PatternMatchingEventHandler)
        :
57  def __init__(self,repo_name, repo_path):
58  print("shitlyfe")
59  print(self)
60  to_ignore = '*/'+repo_name
61  super(EventHandler, self).__init__(
        ignore_patterns=["*/.git/*", to_ignore])
62  self.repo_name = repo_name
63  self.repo_path = repo_path
64  self.repository = ar(repo_path,repo_name)
65
66
67  def on_any_event(self, event):
68  eventType = ["deleted", "modified"]
69  global flag_checkout
70
71  if event.event_type in eventType:
72  pathSplit = event.src_path.split("/")
73  if len(pathSplit) > 4:
74  if ".git" not in pathSplit:
75  print(event.event_type)
76  print(event.src_path)
77  print(pathSplit)
78  print(len(pathSplit))
79  print(self.repo_name)
80  if self.repository.check_repo():
81  self.repository.checkout_to_branch()
82  self.repository.push()
83  else:
84  self.repository.create_branch()
85  self.repository.push()
```

Kode Sumber B.1: Kode sumber Observer.py

Kode Sumber ApplicationRepo.py

```
import datetime
import collections
import requests
from git import Repo, Git
from app import var

class ApplicationRepo():
def __init__(self,repo_path,repo_name):
Repo.init(repo_path)
self.repo = Repo(repo_path)
self.repo_path = repo_path
self.repoName = repo_name
self.api_endpoint = "http://localhost:3000/api/v1/user/repos"
self.api_token = var.API_TOKEN
# self.api_token = "token 5b297d2d0f6e7c6f5d7a7a8de53a776ae008c386"
self.repo_url = var.URL_GITEA+self.repoName+".git"
# self.repo_url = "http://didin:didin23@localhost:3000/didin/"+self.repoName+".git"

# def init_repo(self):
#     Repo.init(self.repo_path)

def create_gitea_repo(self):
data_create = {
"auto_init": True,
"description": "Readme for device "+ self.repoName,

```

```
    "issue_labels": "string_labels",
    "name": self.repoName,
    "private": False
    }

    requests.post(url= self.api_endpoint, headers={
        'Authorization' : self.api_token}, data=
        data_create )
    g=self.repo.git
    g.remote('add', 'origin', self.repo_url)



    def get_branches(self):
    branch = self.repo.git.branch()
    branch = branch.replace('*','')
    branch = branch.replace(' ','')
    branch_split = branch.split('\n')
    indices = [i for i, s in enumerate(branch_split
        ) if 'HEAD' in s]
    if indices:
    del branch_split[indices[0]]
    return branch_split

    def get_hash_branches(self):
    show_ref = self.repo.git.show_ref(hash=10)
    show_ref = show_ref.split('\n')
    return show_ref

    def get_head(self):
    repo = self.repo
    commit = repo.head.commit
    commit = repo.git.rev_parse(commit,short=10)
```

```
57    return commit
58
59    def check_repo(self):
60    head = self.get_head()
61    if head in self.get_hash_branches():
62    return True
63    else:
64    return False
65
66    def checkout_to_branch(self):
67    g = self.repo.git()
68    branch_index = self.get_hash_branches().index(
          self.get_head())
69    branch = self.get_branches()[branch_index]
70    g.checkout(branch)
71
72    def push(self):
73    try:
74    dateNow = datetime.datetime.now()
75    self.repo.git.add('.')
76    self.repo.index.commit(dateNow.strftime("%Y-%m
          -%d %H-%M-%S"))
77    origin = self.repo.remote(name='origin')
78    current_commit = self.get_head()
79    index_branch = self.get_hash_branches().index(
          current_commit)
80    current_branch = self.get_branches()[
          index_branch]
81
82    origin.push(current_branch)
83    print(self.repo)
84    except Exception as e:
85    print("Error Occured")
```

```
86    print(e)
87    finally:
88    print("Push completed")
89
90    print("push function are called")
91
92    def pull(self):
93    g=self.repo.git
94    g.remote('add', 'origin', self.repo_url)
95    self.repo.git.pull('origin', 'master')
96
97    def create_branch(self):
98    g = self.repo.git
99    num = str(len(self.get_branches()))
100   new_branch = num+'-branch'
101   g.checkout(b=new_branch)
102
103   def checkout(self, commit):
104   g = self.repo.git
105   g.checkout(commit)
106
107   def get_log(self):
108   g = self.repo.git
109   return g.log(all=True, oneline=True, graph=True
          )
110
111   def get_list_commits(self):
112   branches = self.get_branches()
113   result = collections.defaultdict(dict)
114   for i in branches:
115   commits = list(self.repo.iter_commits(i))
116   print(i)
117   array_commits = []
```

```
118     for commit in reversed(commits):
119     short_sha = self.repo.git.rev_parse(commit.
            hexsha,short=10)
120     if short_sha == self.get_head():
121     short_sha = short_sha+' {HEAD}'
122     print(short_sha)
123     array_commits.append(short_sha)
124     result[i]=array_commits
125 
126     return result
```

Kode Sumber B.2: Kode sumber ApplicationRepo.py

BIODATA PENULIS

Muhammad Faris Didin Andiyar, biasa dipanggil didin dan lahir pada 23 April 1997 di Mojokerto. Penulis adalah mahasiswa yang sedang menjalani studi di Departemen Informatika Institut Teknologi Sepuluh Nopember. Selama menempuh studi penulis aktif di Organisasi sebagai Ketua Departemen Kaderisasi dan Pemetaan (KDPM) Himpunan Mahasiswa Teknik Computer-Informatika pada tahun ke-3.Pernah menjadi staff di kepanitiaan Schematics. Penulis juga merupakan administrator lab Arsitektur dan Jaringan Komputer (AJK) dan juga pernah menjadi asisten mata kuliah Sistem Operasi dan Jaringan Komputer.