APLIKASI OTOMATISASI JARINGAN BERBASIS WEB MENGGUNAKAN *ANSIBLE*

SKRIPSI

Diajukan sebagai salah satu syarat untuk menyelesaikan Pendidikan Diploma Empat (D-4) Program Studi Teknik Komputer dan Jaringan Jurusan Teknik Elektro Politeknik Negeri Ujung Pandang

MUH. TAUFIK WITRI
425 19 041

PROGRAM STUDI D-4 TEKNIK KOMPUTER DAN JARINGAN
JURUSAN TEKNIK ELEKTRO
POLITEKNIK NEGERI UJUNG PANDANG
MAKASSAR
2023

HALAMAN PENGESAHAN

Skripsi ini dengan judul **Aplikasi Otomatisasi Jaringan Berbasis Web Menggunakan *Ansible*** oleh **Muh. Taufik Witri** NIM 425 19 041 telah diterima dan disahkan sebagai salah satu syarat untuk memperoleh gelar Diploma IV (D-4/S1 Terapan) pada Program Studi Teknik Komputer dan Jaringan Jurusan Teknik Elektro Politeknik Negeri Ujung Pandang

Makassar, 22 Agustus 2023

Mengesahkan,

Pembimbing I

Drs.Kasim, M.T.
NIP. 19630620 1991031 002

Pembimbing II,

Rini Nur,S.T., M.T.
NIP. 197307132009122001

Mengetahui,

Ketua Program Studi
Teknik Komputer dan Jaringan
Politeknik Negeri Ujung Pandang

Eddy Tungaddi, S.T.,M.T.
NIP. 19790823 201012 1 001

HALAMAN PENERIMAAN

Pada hari ini, Selasa tanggal 22 Agustus 2023 Tim Penguji Ujian Sidang Skripsi telah menerima dengan baik skripsi oleh mahasiswa **Muh. Taufik Witri** nomor induk mahasiswa **42519041** dengan judul "**Aplikasi Otomatisasi Jaringan Berbasis Web Menggunakan *Ansible* **"

Makassar, 22 Agustus 2023

Tim Penguji Ujian Sidang Skripsi:

1. Ir.Dahlia,M.T. — Ketua (.....)
2. Meylanie Olivya,S.T.,M.T. — Sekretaris (.....)
3. Prof.Irfan Syamsuddin,S.T. M.Com.ISM.,Ph.D. — Anggota (.....)
4. Irmawati,S.T.,M.T. — Anggota (.....)
5. Drs.Kasim, M.T. — Anggota (.....)
6. Rini Nur,S.T., M.T. — Anggota (.....)

KATA PENGANTAR

Alhamdulillah puji syukur atas segala Rahmat, hidayah dan karunia Allah SWT yang tak tehingga, sehingga penulis mampu menyelesaikan skripsi ini dengan baik. Sholawat serta salam kepada Rasulullah Shallallahu Alaihi Wasallam yang senantiasa menjadi sumber inspirasi dan teladan terbaik bagi umat manusia.

Sebagai salah satu syarat untuk menyelesaikan studi serta memperoleh gelar diploma IV (D-4/S1 Terapan) pada Program Studi Teknik Komputer dan Jaringan Jurusan Teknik Elektro Politeknik Negeri Ujung Pandang, maka skripsi ini disusun dengan sebaik-baiknya. Penulis menyadari bahwa skripsi ini tidak mungkin terselesaikan tanpa adanya, dukungan, bantuan, bimbingan dan doa dari berbagai pihak selama proses penyususan skripsi ini. Pada kesempatan ini penulis menyampaikan terima kasih sebesar-besarnya kepada:

1. Orang tua penulis yakni Bapak Witri Mada dan Ibu Nurhayati yang selalu memberikan dukungan, kasih sayang, semangat dan doa terbaik yang tidak pernah putus sehingga penulis mampu menyelesaikan skripsi ini dengan sangat baik.
2. Bapak Prof. Ir. Ilyas Mansur, M.T selaku Direktur Politeknik Negeri Ujung Pandang.
3. Bapak Ahmad Rizal Sultan, S.T., M.T., Ph.D selaku Ketua Jurusan Teknik Elektro Politeknik Negeri Ujung Pandang.
4. Bapak Eddy Tungadi, S.T., M.T. selaku Koordinator Program Studi Teknik Komputer dan Jaringan.
5. Bapak Drs.Kasim, M.T.selaku pembimbing I dan ibu Rini Nur, S.T., M.T selaku pembimbing II atas segala ilmu, motivasi, nasihat, arahan, bimbingan, bantuan dan kesedian waktu serta kesabarannya dalam membimbing penulis hingga dapat menyelesaikan penelitian ini.
6. Seluruh Dosen dan Staf Jurusan Teknik Elektro, khususnya Program Studi D4 Teknik Komputer dan Jaringan.

7. Teman-teman seperjuangan di Program Studi Teknik Komputer dan Jaringan Angkatan 2019 terutama dari kelas B yang telah belajar dan berjuang bersama selama 4 tahun serta memberikan pengalaman yang terbaik kepada penulis baik dari segi akademik maupun non akademik.
8. Nuraeni sebagai patner peneliti yang membantu peneliti dalam berdiskusi, dan dukungan kepada peneliti.
9. Semua pihak yang telah memberikan bantuan moril maupun materil yang tidak dapat penulis sebutkan satu-persatu.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun demi perbaikan di masa mendatang. Semoga penelitian pada skripsi ini dapat bermanfaat.

Makassar, 16 Agustus 2023

Penulis

DAFTAR ISI

JUDUL i
HALAMAN PENGESAHAN ii
HALAMAN PENERIMAAN iii
KATA PENGANTAR iv
DAFTAR ISI vi
DAFTAR GAMBAR viii
DAFTAR TABEL xi
SURAT PERNYATAAN xii
RINGKASAN xiii
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Tujuan Penelitian 3
1.4 Manfaat Penelitian 3
1.5 Ruang Lingkup Penelitian 4
BAB II TINJAUAN PUSTAKA 5
2.1 *Network Automation* 5
2.2 *Ansible* 5
2.3 *Server* 6
2.4 *Flask* 11
2.4.1 *Flask* 12
2.4.2 Paramiko 13
2.5 Proxmox VE 14
2.6 Virtualbox 16
2.7 *Secure Shell (SSH)* 18
2.8 *Black Box Testing* 18
BAB III METODOLOGI PENELITIAN 19
3.1 Waktu dan Tempat Penelitian 19

3.2 Analisis Kebutuhan 19
3.2.1 Perangkat Keras 19
3.2.2 *Virtual Machine* 20
3.2.3 Perangkat Lunak 20
3.3 Prosedur Penelitian 21
3.3.1 Studi Literatur 21
3.3.2 Analisa Kebutuhan 21
3.3.3 Desain 22
3.3.4 Implementasi 27
3.3.5 Pengujian 27
BAB IV HASIL DAN PEMBAHASAN 29
4.1 Hasil Pembuatan Virtual Machine dan Web Interface 29
4.1.1 PembuatanVirtual Machine 29
4.1.2 Konfigurasi Virtual Mesin 30
4.1.3 Desain UI (*User Interface*) 33
4.2 Konfigurasi *Ansible* pada Master *Server* 37
4.2.1 Konfigurasi Host pada setiap *server* 37
4.2.2 Tes Remot Tanpa password 39
4.2.3 Membuat Program *Ansible* 41
4.3 Implementasi Web *Server* 44
4.4 Proses Penggunaan Aplikasi 51
4.5 Pengujian Sistem 58
4.5.1 Pengujian *White Box* 58
4.5.2 Pengujian Fungsional 64
BAB V PENUTUP 68
5.1 Kesimpulan 68
5.2 Saran 68
DAFTAR PUSTAKA 69

DAFTAR GAMBAR

Gambar 2. 1 Topologi *Ansible* 6
Gambar 2. 2 contoh program Paramiko 14
Gambar 3. 1 Metode Waterfall 21
Gambar 3. 2 Desain Jaringan Model Sistem 22
Gambar 3. 3 Model Perangkat Lunak Master Server 23
Gambar 3. 4 Model Perangkat Lunak Node *Master* 23
Gambar 3. 5 Model Perangkat Lunak Web *Server* 24
Gambar 3. 6 Desain sistem 25
Gambar 3. 7 Activity Diagram 26
Gambar 3. 8 Flowchart 27
Gambar 3. 9 Desain Pengujian Perbandingan Waktu Instalasi 28
Gambar 4. 1 Virtual Mesin pada virtualbox 29
Gambar 4. 2 konfigurasi ip address static dan dinamis 30
Gambar 4. 3 Instal *Flask* 32
Gambar 4. 4 Install *Ansible* 32
Gambar 4. 5 Tampilan Login 33
Gambar 4. 6 Tampilan *Dashboard* 34
Gambar 4. 7 Halaman Form Install 35
Gambar 4. 8 Halaman Form *Uninstall* 35
Gambar 4. 9 Halaman *Services* 36
Gambar 4. 10 Halaman List *Inventory* dan *Node*s 37
Gambar 4. 11 File Host 38
Gambar 4. 12 Hasil Ping Ke *node* 1 38
Gambar 4. 13 hasil Ping Ke *node* 2 38
Gambar 4. 14 hasil Ping Memakai Hostname 39
Gambar 4. 15 Hasil Sukses Remot *node*1 39
Gambar 4. 16 Hasil *Remote node* 2 40
Gambar 4. 17 Program Nginx *Ansible* 41

Gambar 4. 18 Program Uninstal Nginx 42
Gambar 4. 19 Program Instal Apache2 *Ansible* 42
Gambar 4. 20 Program *Uninstall* Apache2 *Ansible* 43
Gambar 4. 21 Program Instal Mysql 43
Gambar 4. 22 Program Uninstal Mysql 44
Gambar 4. 23 Struktur Program Web 44
Gambar 4. 24 Funtion Proses Login 45
Gambar 4. 25 Kode Untuk Menampilkan Halaman *Dashboard* 46
Gambar 4. 26 Kode Proses Instal 46
Gambar 4. 27 Proses Uninstal 47
Gambar 4. 28 Kode Menampikan *Service* 48
Gambar 4. 29 Kode *Inventory* 49
Gambar 4. 30 Kode *Node*s 50
Gambar 4. 31 Halaman Login 51
Gambar 4. 32 Halaman *Dashboard* 51
Gambar 4. 33 Halaman *Service* 52
Gambar 4. 34 Testing aplikasi pada web 52
Gambar 4. 35 Halaman *Service* setelah Install 53
Gambar 4. 36 Testing Nginx 53
Gambar 4. 37 Halaman *Form Uninstal* 54
Gambar 4. 38 halaman *Service* Setelah Uninstal 54
Gambar 4. 39 Testing Nginx 55
Gambar 4. 40 Instal Apache2 55
Gambar 4. 41 Hasil Setelah Instal Apache 56
Gambar 4. 42 Hasil Testing Apache2 56
Gambar 4. 43 Uninstal Apache2 57
Gambar 4. 44 Hasil Setelah Uninstal Apache2 57
Gambar 4. 45 Testing Halaman web Apache2 pada browser 58
Gambar 5. 1 Hasil Setelah Login dengan *Server* Hidup 59
Gambar 5. 2 Hasil Setelah Login dengan *Server* Mati 59
Gambar 5. 3 Penambahan Kode Handling Error 60

Gambar 5. 4 Gambar Setelah Kode ditambahkan 60
Gambar 5. 5 *Node* dalam Keadaan Hidup 61
Gambar 5. 6 Hasil Ketika *Node* Mati 61
Gambar 5. 7 Tambahan Kode 62
Gambar 5. 8 Hasil setelah Kode ditambahkan 62
Gambar 5. 9 Form Instal Diisi kosong 62
Gambar 5. 10 Hasil Ketika Form Tidak Diisi 63
Gambar 5. 11 Kode Tambahan 63
Gambar 5. 12 Hasil Ketika Kode Handling Sudah ditambahkan 64

DAFTAR TABEL

Tabel 3. 1 Perangkat Keras 19
Tabel 3. 2 Spesifikasi *Virtual Machine* 20
Tabel 4. 1 Pengujian *Login* 64
Tabel 4. 2 Pengujian Fitur Aplikasi 65
Tabel 4. 3 Pengujian Koordinasi 66
Tabel 4. 4 Tabel Pengujian Perbandingan Instalasi 67

SURAT PERNYATAAN

Saya yang bertanda tangan dibawah ini:

Nama : Muh. Taufik Witri

NIM : 42519041

Menyatakan dengan sebenar-benarnya bahwa segala pernyataan dalam skripsi ini yang berjudul "Aplikasi Otomatisasi Jaringan Berbasis Web Menggunakan *Ansible*" merupakan gagasan dan hasil karya saya sendiri dengan arahan komisi pembimbing, dan belum pernah diajukan dalam bentuk apapun pada perguruan tinggi dan instansi manapun.

Semua data dan informasi yang digunakan telah dinyatakan secara jelas dan dapat diperiksa kebenarannya. Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan dari penulis lain telah disebutkan dalam naskah dan dicantumkan dalam skripsi ini.

Jika pernyataan saya tersebut diatas tidak benar, saya siap menanggung resiko yang ditetapkan oleh Politeknik Negeri Ujung Pandang.

Makassar, Agustus 2023

Muh. Taufik Witri

NIM. 42519041

APLIKASI OTOMATISASI JARINGAN BERBASIS WEB

MENGGUNAKAN *ANSIBLE*

RINGKASAN

Aplikasi otomatisasi jaringan berbasis web menggunakan Ansible merupakan sebuah solusi inovatif yang memanfaatkan Ansible, sebuah platform open-source untuk otomatisasi tugas-tugas administratif pada jaringan. Aplikasi ini memungkinkan pengelola jaringan untuk secara efisien dan efektif mengelola konfigurasi, monitoring, dan pemeliharaan jaringan mereka melalui antarmuka web yang intuitif.

Penelitian ini bertujuan untuk merancang sebuah aplikasi otomatisasi jaringan menggunakan Ansible berbasis web. Aplikasi ini menggunakan framework Flask sebagai antarmuka webnya dan library paramiko sebagai penghubung antara web dan server. Metode pembuatan aplikasi ini menggunakan pendekatan Waterfall. Pengujian aplikasi ini meliputi tiga jenis uji coba: uji coba whitebox untuk menguji kode dan penanganan kesalahan, uji coba blackbox untuk memastikan semua fitur aplikasi berjalan dengan baik, dan uji coba perbandingan waktu instalasi menggunakan aplikasi dengan cara manual. Uji coba terakhir bertujuan untuk memberikan rekomendasi terkait penggunaan aplikasi. Hasil dari penelitian ini diharapkan dapat memberikan kontribusi signifikan dalam mengelola server di berbagai instansi atau organisasi.

Kata Kunci : Ansible, Aplikasi, Server, Konfigurasi.

BAB I PENDAHULUAN

1.1 Latar Belakang

Terdapat banyak kendala yang biasanya terjadi di kalangan administrator *server* perusahaan. Salah satu contohnya, para administrator memegang banyak *server* untuk tiap orangnya sehingga membuat administrator yang kurang berpengalaman akan kesulitan dalam manajemenya (Khumaidi, 2021).

Server adalah salah satu perangkat jaringan yang harus dikonfigurasi sebelum dipakai. Kebanyakan di perusahaan, sebuah *server* dikonfigurasi secara manual dengan memasukkan perintah secara berulang-ulang sehingga waktu yang diperlukan untuk konfigurasi lama. Dalam konfigurasi *server* juga biasanya terdapat kesalahan-kesalahan konfigurasi yang dimasukkan oleh administrator. Biasanya kesalahan-kesalahan tersebut ditutupi dengan menambah jumlah karyawan, namun hal ini hanya akan menambah pengeluaran biaya untuk menambah karyawan (Khumaidi, 2021).

Solusi yang dapat mengatasi masalah ini adalah menerapkan *Secure Shell* atau SSH pada setiap *server*, sehingga memungkinkan konfigurasi *server* dilakukan tanpa perlu melakukan perpindahan *server*. SSH merupakan protokol administrasi yang memungkinkan pengguna untuk mengakses dan memodifikasi berbagai pengaturan dan file di dalam *server* secara aman. Namun, kendala muncul ketika terdapat banyak konfigurasi yang perlu dimasukkan ke setiap *server* (Affandi et al., 2020).

Otomatisasi jaringan (*Network Automation*) adalah proses mengotomatiskan konfigurasi, pengelolaan, pengujian, penggelaran, dan pengoperasian perangkat fisik dan virtual dalam jaringan. Tugas dan fungsi jaringan sehari-hari yang otomatis dan proses berulang yang dikontrol dan dikelola secara otomatis, ketersediaan layanan jaringan meningkat.

Semua jenis jaringan dapat menggunakan otomatisasi jaringan. Solusi berbasis perangkat keras dan perangkat lunak memungkinkan pusat data, penyedia layanan, dan perusahaan/instansi menerapkan otomatisasi jaringan untuk meningkatkan efisiensi, mengurangi kesalahan manusia, dan menurunkan biaya operasional.

Penelitian tentang otomatisasi jaringan menggunakan *Ansible* pernah diterapkan. *Ansible* adalah sebuah alat *open-source* untuk pengaturan perangkat lunak, pengelolaan konfigurasi, dan penerapan aplikasi. Namun, pada penelitian tersebut, penggunaan *Ansible* masih terbatas sebagai *tool backend engineer* saja dan untuk itu masih diperlukan pembuatan perintah secara manual melalui *command line Linux*. Keterbatasan ini mengindikasikan bahwa pemanfaatan *Ansible* belum optimal dan belum dapat diakses dengan mudah oleh berbagai pihak.

Penggunaan *Ansible* ke dalam aplikasi web, dapat menjadi alternatif hal tersebut diatas. Pendekatan ini, dapat mengatasi kesenjangan antara penggunaan manual *Ansible* dan penggunaan yang lebih meluas dan memudahkan akses dan pemanfaatan fitur-fitur *Ansible* melalui antarmuka web, menghilangkan kebutuhan untuk membuat perintah *Ansible-playbook* secara manual melalui baris perintah.

Pada penelitian ini diimplementasikan sebuah aplikasi web berbasis *Ansible*

menggunakan bahasa pemprograman *Flask* dimana *Flask* sebagai *framework*/kerangka kerja mikro dalam membangun web dengan *Flask* tersebut. Metode SSH dalam aplikasi ini untuk mengakses perangkat *node* dari *server* menggunakan Paramiko sebagai *library* pada *Flask* (Khumaidi, 2021) (Affandi et al., 2020).

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, rumusan masalah yang didapatkan adalah "Bagaimana mengimplentasikan aplikasi web menggunakan *Ansible* untuk otomatisasi jaringan?"

1.3 Tujuan Penelitian

Tujuan yang diperoleh dalam penelitian ini adalah mengimplentasikan aplikasi web menggunakan *Ansible* untuk otomatisasi jaringan

1.4 Manfaat Penelitian

Beberapa manfaat yang diperoleh dalam penelitian ini, yaitu:

1) Membantu para administrator dalam konfigurasi *server*
2) Efisiensi waktu konfigurasi *server*.
3) Membantu para administrator pemula

1.5 Ruang Lingkup Penelitian

Beberapa ruang lingkup penelitian ini,yaitu:

1) Sistem operasi yang dipakai pada *server* adalah Linux
2) Otomatisasi ini hanya menggunakan Platform *Ansible*
3) Fitur *Network Automation* yang diimplementasikan pada aplikasi adalah monitoring *node*, monitoring *service* dan instalasi *service* pada *node*.

BAB II TINJAUAN PUSTAKA

2.1 *Network Automation*

Network Automation merupakan sebuah solusi dalam mengimplementasikan beberapa pekerjaan yang rumit atau berulang. Dalam metode tradisional konfigurasi pada perangkat jaringan dilakukan dengan masuk ke dalam perangkat jaringan untuk mengkonfigurasi perangkat jaringan tersebut. Pekerjaan tersebut terlihat mudah bila yang dikonfigurasi masih satu atau dua perangkat jaringan, hal ini akan terlihat rumit jika terdapat banyak perangkat jaringan seperti *router* dan *server* yang harus dikonfigurasi.

Network Automation juga adalah proses mengotomatisasi konfigurasi, pengujian, operasi perangkat virtual dalam jaringan, dan manajemen. Dalam implementasi sistem otomatisasi jaringan dibutuhkan sebuah software atau program untuk menjalankan perintah ke dalam sebuah perangkat. Salah satu program yang dapat digunakan adalah *Flask*. *Flask* sendiri memiliki beberapa jenis *library* yang digunakan untuk implementasi sistem otomatisasi jaringan diantaranya adalah Paramiko dan *Netmiko* (Fahmi et al., 2021).

2.2 *Ansible*

Ansible merupakan sebuah *tool Automation* yang dapat digunakan untuk mempermudah kita untuk melakukan instalasi, *deployment*, *provisioning server*. *Ansible* berguna untuk mempermudah pekerjaan mengatur dan mengelolah *server* yang memiliki jumlah yang sangat banyak, dengan beragam aplikasi atau *software* di dalamnya. *Ansible* terkenal dengan kemudahan membaca konfigurasinya, cara

kerja *Ansible* itu hanya dengan menggunakan koneksi SSH pada tiap *server* yang akan di jalankan *script Ansible* ini (Yudha, 2020).

Ansible menggunakan SSH untuk terkoneksi ke *server* dan menjalankan semua instruksi secara *remote*, tentu hal ini sama halnya dengan yang biasa kita lakukan saat me-*remote server* dan lebih kerennya lagi, kita tidak perlu memasang Agen di setiap *server*, kita hanya perlu memasang *Ansible* di mana *Ansible* dijalankan. Topologi *Ansible* dapat dilihat pada gambar 2.1

Gambar 2. 1 *Topologi Ansible*

Pada penelitian sebelumnya *Ansible* ini dikonfigurasi menggunakan *command line* dan dieksekusi secara langsung pada master *server*. Selain itu Ansible juga digunakann untuk konfigurasi beberapa perangkat jaringan dengan cara manual yaitu, dengan menuliskan kode dalam yaml dan mengeksekusinya dengan perintah ansible-playbook. Pada penelitian sebelumnya juga menjelaskan bahwa Ansible lebih cepat dari Docker karena ansible menggunakan tools SSH untuk menjalankannya(Yudha, 2020). (Affandi et al., 2020).

2.3 *Server*

Server adalah suatu sistem komputer yang memiliki layanan khusus berupa penyimpanan data. *Server* akan menyimpan beragam jenis dokumen dan

menyediakan informasi untuk pengguna atau pengunjungnya.

Secara umum, fungsi utama *server* adalah melayani dan bertanggung jawab penuh terhadap permintaan data dari komputer klien. Selain itu, fungsi *server* juga untuk mengatur hak akses ke dalam jaringan yang bisa digunakan oleh komputer klien. Di samping itu, komputer sering juga berisi berbagai data informasi, di mana *server* tersebut memiliki tugas memberikan layanan bagi para klien yang terhubung dengannya.

Server adalah sistem komputer yang memiliki layanan khusus berupa penyimpanan data. *Server* akan menyimpan beragam jenis dokumen dan menyediakan informasi untuk pengguna atau pengunjungnya. *Server* berperan penting dalam menyediakan akses lebih cepat untuk mengirim atau menerima data yang tersedia pada *server*. Dalam bentuk fisiknya, *server* berwujud jaringan komputer dan memiliki ukuran yang sangat besar dengan beberapa komponen pendukung prosesor dan RAM yang berkapasitas besar.

Berikut ini beberapa fungsi *server*, yaitu:

1. Melayani Permintaan Komputer Klien

Fungsi utama dari *server* adalah akan melayani segala permintaan dari klien untuk diproses. Baik itu permintaan data atau aplikasi untuk dijalankan oleh klien. Untuk mendukung fungsi tersebut, *server* biasanya menggunakan sistem operasi yang cepat dan aman. Jadi, klien dapat bekerja lebih efektif dan tentunya aman.

2. Menyimpan Data atau Informasi

Fungsi lainnya dari *server* adalah sebagai tempat penyimpanan data yang

dikirimkan dari klien. Data yang tersimpan tersebut dapat berupa jenis dokumen dan informasi yang kompleks. Untuk bisa menampung data yang banyak, *server* tentu harus memiliki kapasitas yang besar. Sehingga, klien bisa menyimpan dan mengakses data bersama dengan klien lainnya.

3. Menyediakan *Database* untuk Dijalankan

Server juga memiliki fungsi menyediakan *database* sebagai penyimpan dan pengolah data. Biasanya perusahaan besar memanfaatkan fungsi ini untuk menerapkan big data. Nantinya, semua data yang tersimpan dalam *database* dapat diolah dan diakses oleh pengguna. Dengan adanya layanan ini, banyak perusahaan dapat mengembangkan produk bisnisnya.

4. Mengatur Lalu Lintas Transfer Data atau File

Fungsi *server* yang lainnya yaitu akan mengatur komunikasi dan transfer informasi pada klien. Bisa dibayangkan bagaimana sibuknya *server* ketika banyak klien yang akan mengajukan permintaan. Untuk itu, perangkat *server* biasanya memiliki kapasitas seperti hardisk dan RAM yang tinggi.

5. Mengamankan dari Serangan Kejahatan

Fungsi *server* yang terakhir adalah untuk melindungi komputer atau website dari serangan hacker. Setiap ada request data dari klien, *server* akan mengecek alamat IP dan informasi lainnya. Jika terdapat hal yang mencurigakan, seperti ancaman malware, *server* dapat mencegah akses alamat IP tersebut. Dengan begitu, data-data yang tersimpan pada komputer atau website dapat tetap aman (Mustofa, 2021).

Setelah mengetahui apa itu *server* dan fungsinya, berikut ini ada beberapa

manfaat yang di dapat dari *server*, yaitu:

1. Menjamin Keamanan Data

Dengan adanya *server*, maka anda dapat menjaga dan menyimpan seluruh data anda dengan lebih aman dan ter-*monitoring* setiap saat. Selain itu, data yang telah anda simpan dapat diakses dengan cepat sesuai dengan kebutuhan anda. Dan yang paling penting, anda juga dapat menghemat penyimpanan data dari perangkat komputer klien anda dan dapat dialihkan menuju *server*.

2. Menghemat Biaya

Manfaat yang selanjutnya tentu saja dapat menghemat biaya pengeluaran untuk membeli kebutuhan perangkat penyimpanan data. *Server* telah menyediakan kapasitas penyimpanan yang lebih besar daripada komputer klien pada umumnya.

3. Memudahkan dalam Manajemen Data

Manfaat yang lainnya yaitu dapat memudahkan dalam proses pengelolaan berbagai dokumen dan informasi penting yang membutuhkan penanganan secara cepat. *Server* juga menjamin penyimpanan data dalam jangka waktu yang lebih panjang. Saat ini juga tersedia penyimpanan *server* berbasis *cloud* atau awan. Data anda akan tersimpan di internet dengan alokasi penyimpanan yang jauh lebih besar lagi.

4. Mengoptimalkan Fungsi Kolaborasi Antar Tim

Manfaat yang terakhir yaitu mampu untuk memaksimalkan kolaborasi dan kerja antar setiap tim atau departemen. Di mana, dalam sebuah bisnis terutama dalam bidang IT mengharuskan setiap tim untuk bekerja tidak hanya secara *offline* saja. Namun, dituntut untuk dapat bekerja secara *online* atau *remote*. Sehingga,

server sangat berperan penting untuk meningkatkan produktivitas karyawan perusahaan atau instansi terkait.

Server terbagi menjadi beberapa jenis-jenis *server*, yaitu:

1. *Mail Server*

Mail server merupakan jaringan komputer yang melayani pengguna dalam bertukar pesan ke sesama pengguna secara elektronik. *Mail server* sendiri dibagi menjadi dua kategori, yaitu *server* surat masuk dan *server* surat keluar. *Server* surat masuk terdiri dari IMAP dan POP3. IMAP *(Internet Message Access Protocol)* akan menyimpan salinan pesan dikirim dan diterima ke dalam *Mail Server*. Sementara itu, *POP3 (Post Office Protocol 3rd version)* adalah protokol pesan elektronik untuk menyimpan pesan dikirim dan diterima PC *Local Hard Drive*. *Mail server* berfungsi untuk bertukar pesan ke sesama pengguna dengan file terdukung berupa gambar, teks, dan lainnya.

2. File *Server*

Salah satu jenis *server* pada komputer yaitu file *server*. Jenis *server* ini merupakan jaringan komputer yang memberikan akses berupa lokasi disk yang berisi gambar, musik, video, dokumen, dan lainnya. File *server* dirancang untuk pengguna memungkinkan dalam penyimpanan dan pengambilan data dengan perhitungan melalui *workstation.*

3. *Database Server*

Database server adalah kumpulan data yang diperoleh untuk kemudian disimpan di dalam komputer. Adapun penggunaan *database* dalam komputer ini ditujukan untuk mempermudah pengguna dalam mengolah data. Sederhananya,

database terdiri dari kumpulan tabel-tabel yang menyimpan data serta informasi. Salah satu fungsi *database* adalah untuk menghindari data ganda yang tersimpan. Suatu software DBMS bisa di-*setting* agar mampu mengenali duplikasi data yang terjadi saat diinput. Hal ini karena sifat *database* dapat diakses oleh lebih dari satu pengguna.

4. Web *Server*

Web *server* adalah jaringan komputer yang melayani khusus permintaan HTTP dan HTTPS. Nantinya, web *server* akan menerima kode dari browser, kemudian mengirimnya kembali dalam bentuk laman web. Dengan kata lain, web tersebut dikirim oleh web *server* dalam bentuk dokumen HTML dan CSS, lalu diproses oleh browser menjadi laman web yang mudah dibaca oleh pengguna (Mustofa, 2021).

2.4 *Flask*

Flask adalah bahasa pemrograman *interpreted* yang artinya program yang dibuat dalam bahasa pemrograman *Flask* dapat dijalankan segera setelah file yang berisi *Flask codes* selesai dibuat. Oleh karena itu aplikasi yang dibuat dalam *Flask* akanlebih mudah diperbaiki jika ada kesalahan dan jadi cepat selesai jika dibandingkan dengan program yang dibuat dengan bahasa pemrograman yang memerlukan *compiling*. Selain itu, *Flask* juga dikenal sebagai bahasa pemrograman yang sangat mudah untuk dipelajari dan didukung oleh komunitas yang kuat dan modul-modul pendukung untuk berbagai bidang.

Flask menjadi bahasa pemrograman yang sangat populer baik bagi pemula maupun yang telah berpengalaman. *Flask* juga mendukung berbagai gaya

pemrograman yang meliputi pemrograman terstruktur dan beroientasi obyek. Seperti telah disinggung di atas, ada banyak modul-modul pendukung yang dapat mempermudah pengembangan suatu program. Untuk keperluan pembuatan grafik atau kurva, ada modul Matplotlib, yang merupakan kumpulan modul pendukung yang lengkap untuk pembuatan visualisasi statis, anamasi dan interaktif dalam *Flask* (Suwito, 2021).

2.4.1 *Flask*

Flask adalah sebuah web *framework* yang ditulis dengan bahasa *Flask* dan tergolong sebagai jenis micro*framework*. *Flask* berfungsi sebagai kerangka kerja aplikasi dan tampilan dari suatu web. Dengan menggunakan *Flask* dan bahasa *Flask*, pengembang dapat membuat sebuah web yang terstruktur dan dapat mengatur behaviour suatu web dengan lebih mudah.

Flask termasuk pada jenis micro*framework* karena tidak memerlukan suatu alat atau pustaka tertentu dalam penggunaannya. Sebagian besar fungsi dan komponen umum seperti validasi form, databse, dan sebagainya tidak terpasang secara default di *Flask*. Dengan begitu, fleksibilitas serta skalabilitas dari *Flask* dapat dikatakan cukup tinggi dibandingkan dengan *framework* lainnya (Djamaluddin, 2021). Berikut fitur – fitur bawaan dari *Flask* diantaranya adalah

1. Built-in development server
2. Debugger cepat
3. Integrated support untuk pengetesan unit
4. Kompatibel dengan mesin aplikasi Google

5. RESTful request dispatching
6. Jinja2 templating
7. Mendukung secure cookies
8. Berbasis unicode
9. Mengikuti WSGI 1.0

Flask juga didukung dengan dokumentasi yang sangat baik dan banyak forum yang ada di internet untuk diskusi terkait masalah *Flask* (Djamaluddin, 2021).

2.4.2 Paramiko

Paramiko adalah sebuah pustaka (*library*) *Flask* yang menyediakan implementasi dari protokol SSH (*Secure Shell*). Paramiko memungkinkan Anda untuk terhubung dengan *server* jarak jauh secara aman melalui jaringan dan melakukan berbagai operasi seperti eksekusi perintah jarak jauh, transfer berkas, dan *tunneling*. Paramiko umumnya digunakan untuk otomatisasi tugas pada *server* jarak jauh, mengelola infrastruktur, dan berinteraksi dengan perangkat jaringan.

1. Fitur utama Paramiko meliputi

Implementasi Protokol SSH: Paramiko memungkinkan Anda untuk menjalin koneksi aman ke *server* jarak jauh menggunakan protokol SSH, yang menyediakan enkripsi dan otentikasi untuk komunikasi data. Eksekusi Perintah Jarak Jauh: Anda dapat menjalankan perintah pada *server* jarak jauh menggunakan kelas SSH*Client* dari Paramiko, yang memungkinkan Anda untuk mengirim perintah dan menerima keluaran perintah tersebut secara programatik.

SFTP (*Secure File Transfer Protocol*): Paramiko menyediakan implementasi klien SFTP yang memungkinkan Anda untuk melakukan transfer berkas yang aman antara sistem lokal dan jarak jauh. Hal ini dapat berguna untuk tugas seperti mengunggah, mengunduh, atau mengelola berkas pada *server* jarak jauh. Pengelolaan Kunci SSH: Paramiko mendukung otentikasi berbasis kunci SSH, yang lebih aman dan nyaman daripada menggunakan kata sandi untuk otentikasi.

Tunneling: Paramiko dapat digunakan untuk membentuk *tunnel* SSH, yang memungkinkan komunikasi aman antara host yang berbeda melalui saluran terenkripsi (Karki & V, 2021).

```python
import paramiko

# Membuat instansi klien SSH
ssh = paramiko.SSHClient()

# Otomatis menambahkan kunci host server (ini tidak aman di produksi)
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

# Terhubung dengan server jarak jauh
ssh.connect('nama_host', username='nama_pengguna', password='kata_sandi')

# Menjalankan perintah jarak jauh
stdin, stdout, stderr = ssh.exec_command('ls -l')

# Menampilkan keluaran perintah
print(stdout.read().decode())

# Menutup koneksi SSH
ssh.close()
```

Gambar 2. 2 contoh program Paramiko

2.5 Proxmox VE

Proxmox VE *(Virtual Environtment)* adalah *distro Linux server* dengan

kode sumber terbuka untuk membangun lingkungan virtualisasi berbasis distribusi *Linux Debian*. Proxmox VE digunakan untuk mengimplementasikan dan mengelola mesin virtual. Proxmox VE menggunakan *kernel* dari *Red Hat Enterprise Linux (RHEL)* yang sudah dimodifikasi. Ada dua teknologi *hypervisor* yang didukung oleh Proxmox VE yaitu *container-based virtualization* dan *Kernel-based Virtual Machine* (KVM).

Beberapa fitur utama yang dimiliki oleh Proxmox VE, diantaranya:

1) *Open source* :

Proxmox VE sepenuhnya open source atau bersifat gratis dibawah *General Public License*, version 3 (GNU AGL, v3) yang artinya dapat digunakan dengan bebas.

2) *High Availability* :

Beberapa *server* yang di-*install* Proxmox VE dapat digabungkan menjadi satu *cluster*. Dalam *mode cluster*, ketika terjadi kegagalan pada salah satu *node* maka *node* lain yang berada pada satu *cluster* akan mem-*backup node* tersebut untuk meminimalisir gangguan layanan.

3) *Flexible storage* :

Memiliki banyak pilihan penyimpanan yang tersedia termasuk penyimpanan lokal dan penyimpanan berbasis jaringan, seperti LVM, iSCSI, NFS, GFS, dan CEPH.

4) *Live migration* :

Live migration memungkinkan untuk memindahkan mesin virtual dari satu *node Proxmox* ke *node Proxmox* lainnya dengan waktu *downtime* yang sangat kecil

atau tanpa adanya *downtime*.

5) *Bridged networking* :

Proxmox VE memungkinkan pengguna untuk membuat jaringan *private* antara mesin virtual. Selain itu, VLAN juga tersedia.

6) *OS Template* :

Proxmox VE memungkinkan pengguna untuk membangun *template* sistem operasi sendiri atau dapat meng-upload file ISO yang sudah dimiliki ke *node Proxmox*.

7) *Scheduled backup* :

Tersedia antarmuka untuk pengguna yang digunakan untuk mengatur strategi *backup*. File backup dapat disimpan secara local atau penyimpanan lain yang sudah dikonfigurasi.

8) *Command-line (CLI) tool* :

Proxmox VE menyediakan cara manajemen lain untuk pengguna mengatur sumber daya mesin virtual dan lainnya.(Vassa, 2021)

2.6 Virtualbox

VirtualBox adalah sebuah perangkat lunak virtualisasi yang memungkinkan Anda untuk menjalankan dan mengelola mesin virtual di dalam sistem operasi utama Anda. Dengan menggunakan VirtualBox, Anda dapat membuat dan menjalankan mesin virtual yang dapat menjalankan sistem operasi tambahan, seperti Linux, Windows, macOS, dan lainnya, di dalam lingkungan yang terisolasi dari sistem utama Anda.Beberapa fitur utama dari VirtualBox meliputi:

1. Pemisahan Lingkungan: VirtualBox memungkinkan Anda untuk menciptakan lingkungan terisolasi yang disebut mesin virtual. Setiap mesin virtual dapat memiliki sistem operasi dan aplikasi sendiri-sendiri, sehingga Anda dapat menguji, mengembangkan, atau menjalankan perangkat lunak dalam lingkungan yang terpisah.
2. Migrasi dan Snapshot: Anda dapat dengan mudah membuat salinan cadangan (snapshot) dari mesin virtual, yang memungkinkan Anda untuk kembali ke titik tertentu dalam waktu jika sesuatu berjalan tidak sesuai. Anda juga dapat memindahkan mesin virtual antara berbagai host VirtualBox dengan relatif mudah.
3. Konfigurasi Jaringan: VirtualBox menyediakan berbagai pilihan untuk mengatur konfigurasi jaringan pada mesin virtual, termasuk modus NAT (Network Address Translation), modus jembatan, dan modus berbasis host.
4. Pengelolaan Sumber Daya: Anda dapat mengatur seberapa banyak memori, CPU, ruang penyimpanan, dan sumber daya lainnya yang dialokasikan untuk setiap mesin virtual.
5. Integrasi Layar Penuh: VirtualBox mendukung mode layar penuh yang memungkinkan mesin virtual tampak seperti sistem operasi fisik.
6. Dukungan untuk USB: Anda dapat mengaktifkan dukungan USB dalam mesin virtual sehingga perangkat USB dapat digunakan dalam lingkungan virtual.

VirtualBox adalah perangkat lunak *open-source* yang dikembangkan oleh Oracle Corporation. Ini tersedia dalam berbagai platform, termasuk Windows, macOS, Linux, dan lainnya. VirtualBox memiliki antarmuka pengguna grafis yang

mudah digunakan serta dukungan untuk mengelola mesin virtual melalui baris perintah (Vassa, 2021).

2.7 *Secure Shell (SSH)*

SSH merupakan singkatan dari *Secure Shell* yang merupakan aplikasi pengganti *remote* login, tidak berbeda jauh dari rsh dan rlogin, dan merupakan sebuah *protocol* jaringan, dengan adanya SSH ini pengguna dapat menukarkan data melalui saluran yang aman antara dua perangkat jaringan. *Protocol* jaringan ini sering digunakan pada sistem operasi Linux dan protocol yang diciptakan Tatu Yloen, seorang peneliti di *Helsinki University of Technology*, Finlandia dirancang sebagai pengganti *Telnet* dan *Shell Remote* taka man lainnya, yang mengirim informasi, terutama kata sandi, dalam bentuk teks yang sederhana sehingga membuatnya mudah untuk dicegat. Enkripsi yang dilakukan oleh SSH menyediakan kerahasiaan dan integritas datanya melalui jaringan tidak aman seperti Internet (Ardiansyah, 2019).

2.8 *Black Box Testing*

Black Box Testing adalah pengujian yang dilakukan untuk mengamati hasil *input* dan *output* dari perangkat lunak tanpa mengetahui struktur kode dari perangkat lunak. Pengujian ini dilakukan di akhir pembuatan perangkat lunak untuk mengetahui apakah perangkat lunak dapat berfungsi dengan baik atau tidak. Keuntungan menggunakan *black box testing* adalah penguji tidak harus memiliki pengetahuan tentang suatu bahasa pemrograman (Setiawan, 2021).

BAB III METODOLOGI PENELITIAN

3.1 Waktu dan Tempat Penelitian

Penelitian ini dilakukan di Laboratorium Jaringan Komputer Program Studi D4 Teknik Komputer dan Jaringan, Jurusan Teknik Elektro, Kampus 1 Politeknik Negeri Ujung Pandang, Jl. Perintis Kemerdekaan Km. 10, Kota Makassar, Sulawesi Selatan dari bulan Februari sampai bulan Juli 2023

3.2 Analisis Kebutuhan

3.2.1 Perangkat Keras

Dalam Penelitian ini perangkat keras yang digunakan, dapat dilihat pada tabel 3.1 :

Tabel 3. 1 Perangkat Keras

No	Perangkat Keras	Deskripsi
1	Laptop/pc	Kebutuhan sistem minimum : Intel Core i3, RAM minimal 4GB dan harddisk minimal 500GB
2	*Server*	Intel core i5, RAM minimal 8GB, Harddisk minimal 500GB

3.2.2 *Virtual Machine*

Virtual machine yang buat ada 3 yaitu satu sebagai master *server* dan lainnya sebagai *node server*. Untuk spesifikasinya master lebih tinggi spesifikasinya daripada *node server*. Untuk lebih jelasnya bisa dilihat pada tabel 3.2

Tabel 3. 2 Spesifikasi *Virtual Machine*

No	Nama	Deskripsi
1	Master *server*	RAM 2 GB, Storage 40GB
2	*Node Server*	RAM 1 GB, Storage 40GB

Spesifikasi *Virtual Machine* ini digunakan untuk desain dan pengujian

3.2.3 Perangkat Lunak

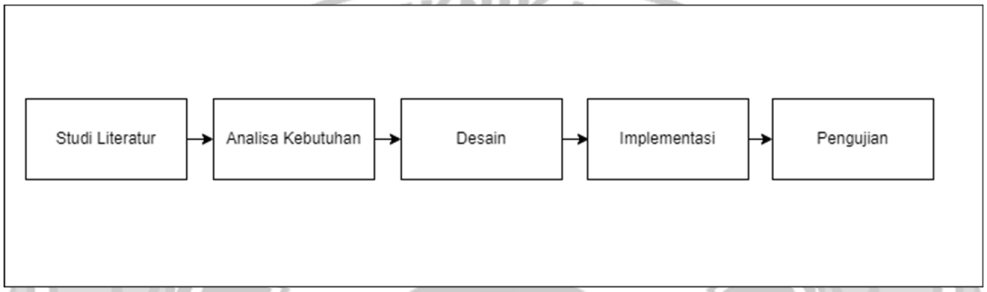
Dalam penelitian ini perangkat lunak yang digunakan, dapat dilihat pada tabel 3.2:

Tabel 3.Perangkat Lunak

No	Perangkat Lunak
1	Sistem Operasi Windows 64 bit
2	*Proxmox*
3	Visual Studio Code
4	Putty
5	Google Chrome
6	*Library Flask*

3.3 Prosedur Penelitian

Penelitian ini memiliki beberapa tahapan yang akan dilalui mulai dari analisis kebutuhan hingga tahap penerapa dan pemeliharaan. Gambar 3.1 menunjukkan metode *waterfall* penelitian yang akan dilakukan:



Gambar 3. 1 *Metode Waterfall*

3.3.1 Studi Literatur

Studi literatur yang dilakukan dalam prosedur penelitian ini dengan menggunakan *Library* Research yang merupakan tahapan untuk pengumpulan data dari beberapa buku, jurnal, skripsi, tesis maupun literatur lainnya yang dapat dijadikan acuan pembahasan dalam masalah ini. Tahap ini akan berkaitan dengan pengumpulan data melalui internet ataupun penelitian-penelitian sebelumnya yang sudah pernah dibuat.

3.3.2 Analisa Kebutuhan

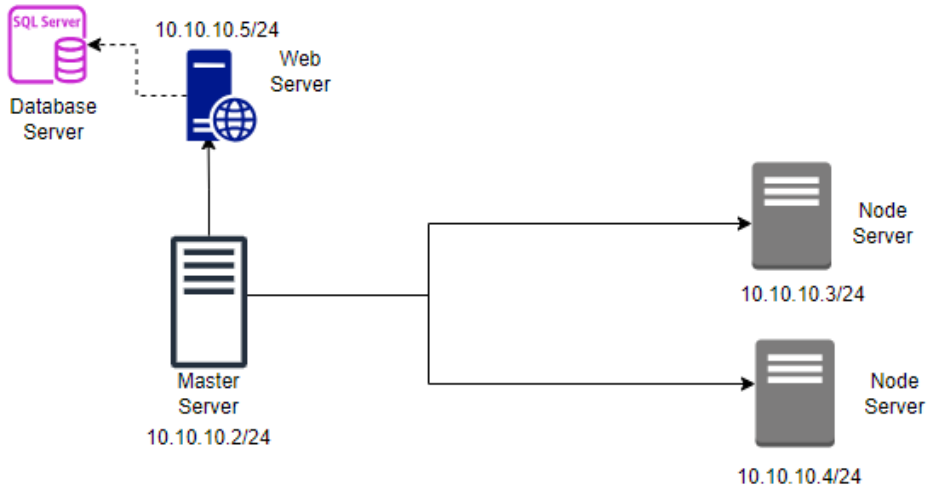
Pengumpulan semua kebutuhan penelitian mulai dari perangkat keras dan perangkat lunak yang akan digunakan pada tahap penulisan kode program.

3.3.3 Desain

Desain dilakukan untuk menggambarkan secara umum sistem yang akan dibuat berdasarkan dari hasil analisa kebutuhan.

a) Desain Jaringan

Dalam perancangan digunakan *server* yang akan bertindak sebagai master dan juga sebagai *node server*. *Node server* ini sendiri adalah beberapa *server* yang telah di *cluster* yang nantinya akan dikontrol melalui master *server*. Gambar 3.2 menunjukkan desain jaringan.



Gambar 3. 2 Desain Jaringan Model Sistem

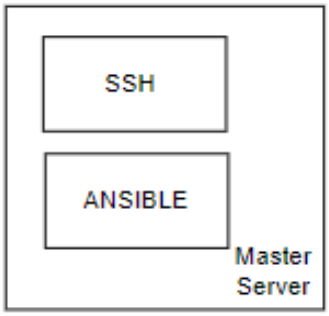
Keterangan:

1) *Master server* digunakan untuk mengontrol *Node Server*
2) *Database server* digunakan untuk menyimpan data
3) *Node Server* digunakan untuk melayaani *service* seperti web, *database* dan lain-lain

4) *Web Server* digunakan sebagai tempat untuk mengakses aplikasi web

a.1) Desain Master *Server*

Desain Master *server* bisa dilihat pada gambar 3.3



Gambar 3. 3 Model Perangkat Lunak Master *Server*

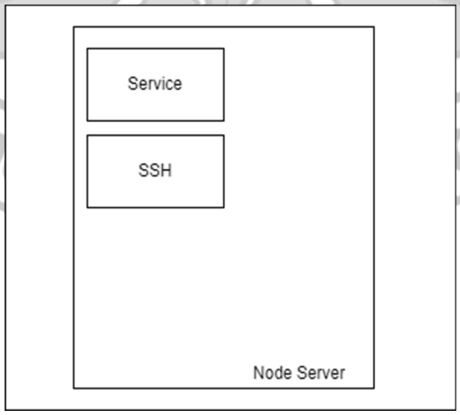
Keterangan:

1) SSH berfungsi untuk *remote* access ke *node*

2) *Ansible* berfungsi sebagai automation

a.2) Desain *Node Server*

Desain *node server* dapat dilihat pada gambar 3.4



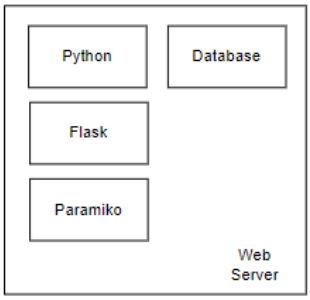
Gambar 3. 4 Model Perangkat Lunak Node *Master*

Keterangan:

1) *Service* adalah Aplikasi *service* yang akan diinstall dan dijalankan pada *Node Server*
2) SSH berfungsi untuk Master bisa melakukan *remote* terhadap *node*

a.3) Desain Web *Server*

Desain Web *Server* dapat diliat pada gambar 3.5



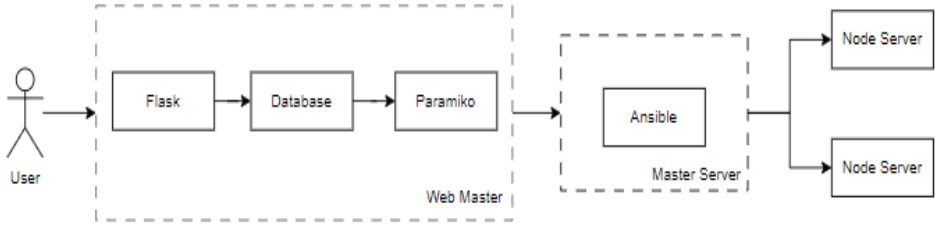
Gambar 3. 5 Model Perangkat Lunak Web *Server*

Keterangan

1) Paramiko berfungsi untuk menghubungkan antara Master dengan *Node* dengan *Flask*
2) *Flask* berfungsi sebagai interface dari Web dari Sistem ini
3) *Database* sebagai tempat penyimpanan data *node*, master dan juga *service*
4) *Python* bahasa pemrograman yang dipakai untuk menjalakan web.

b) Desain Sistem

Desain Sistem merupakan model perancangan sistem yang akan dibuat berdasarkan hasil analisis sistem yang menggambarkan desain sistem yang akan dirancang. Pada gambar 3.6 menunjukkan perancangan desain sistem yang dibuat berdasarkan analisis kebutuhan.



Gambar 3. 6 Desain sistem

Keterangan:

1) User digunakan untuk me-*remote master server*
2) *Flask* digunakan untuk sebagai penyedia web berbasis *Flask* yang akan menampilkan web interface dari *master server*
3) Paramiko digunakan sebagai *library Flask* untuk menghubungkan perangkat jaringan ke web interface
4) *Database* sebagai tempat penyimpanan data *node*, master dan juga *service*
5) *Ansible* berfungsi sebagai alat automation yang akan digunakan
6) Master *server* digunakan untuk mengontrol semua *node server*
7) *Node server* digunakan untuk menyediakan *service*

c) *Activity Diagram*

Activity Diagram merupakan rancangan aliran aktivitas atau aliran kerja dalam sebuah sistem yang akan dijalankan. Gambar 3.7 menampilkan proses *activity diagram* yaitu bagaimana interaksi admin dengan sistem.

Gambar 3. 7 Activity Diagram

d) Flowchart

Flowchart menggambarkan alur dari sistem yang akan dibuat. Untuk alur sistemnya, yaitu *remote master server* dari jauh menggunakan *web*. Jika berhasil me-*remote*, pilih ip *node server* untuk menginstal *service* pada *node server*. Semua *service* yang tersedia akan tampil. *Service* yang tampil bisa dipilih untuk dipasangkan pada *node server* yang telah dipilih.

Gambar alur sistem ditunjukkan pada gambar 3.8

Gambar 3. 8 Flowchart

3.3.4 Implementasi

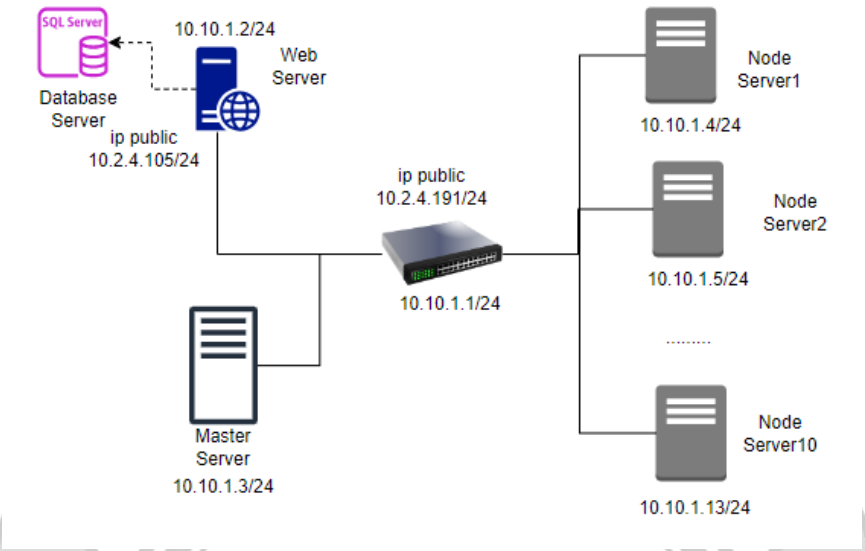
Tahap implementasi adalah tahap mengumpulkan semua kebutuhan mulai dari perangkat keras dan perangkat lunak. Membangun sistem mulai dari penulisan program sampai membuat *Ansible* yang mulanya dikonfigurasi secara manual menjadi sebuah aplikasi berbasis web.

3.3.5 Pengujian

Tahapan pengujian ini dilakukan di Politeknik Negeri Ujung Pandang pada lab CNAP. Pengujian yang dilakukan adalah pengujian *Black Box Testing* dari web yang akan dihasilkan untuk mengontrol master dan *node server*.

Desain Pengujian Perbandingan Lama Waktu Instalasi

Untuk desain pengujian perbandingan waktu instalasi bisa diliat pada gambar 3.9



Gambar 3. 9 Desain Pengujian Perbandingan Waktu Instalasi

Desain pengujian ini memiliki beberapa perangkat seperti:

a. 1 buah Web Server yang disatukan dengan Database Server sebagai tempat aplikasi
b. 1 buah Master Server sebagai control untuk Node Server
c. 1 buah Mikrotik sebagai penghubung antar perangkat
d. 10 buah Node Server sabagai tempat untuk menginstal Service

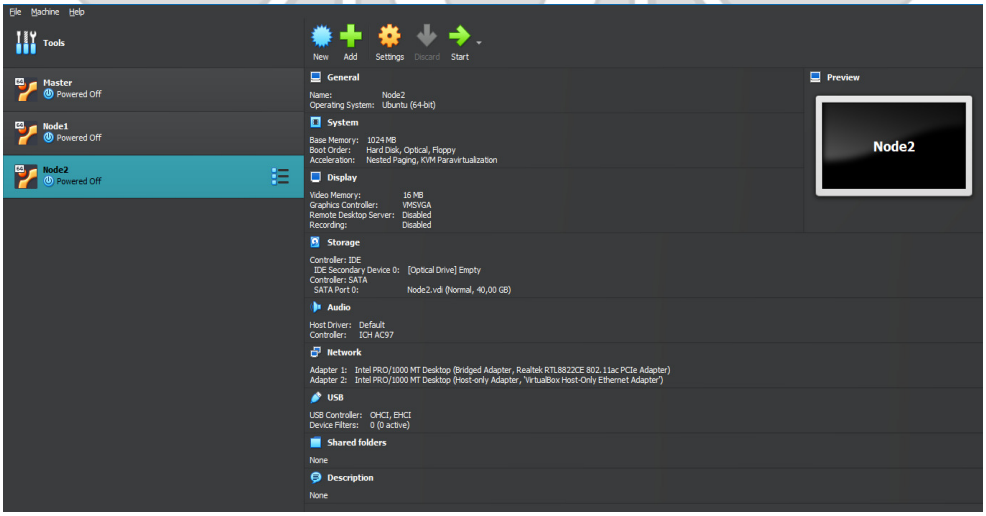
BAB IV HASIL DAN PEMBAHASAN

Hasil yang didapatkan dari penelitian ini berdasarkan rancangan, tahapan–tahapan pelaksanaan, dan pengujian yang sesuai dengan pokok permasalahan dan ruang lingkup penelitian. Penelitian ini berfokus pada pembuatan aplikasi web menggunakan *Flask* dan pembuatan otomatisasi menggunakan *Ansible.*

4.1 Hasil Pembuatan Virtual Machine dan Web Interface

4.1.1 PembuatanVirtual Machine

Virtual machine dibuat sebagai master dan juga *node server* yang akan mengatur *service- service* yang akan dijalankan. Virtual mesin yang digunakan ada 3 salah satunya sebagai master *server* dan lainnya jadi *node server.* Untuk Spesifikasinya master *server* lebih tinggi dari *node server.* Setelah dibuat maka akan muncul tampilan 3 virtual mesin seperti pada gambar 4.1



Gambar 4. 1 Virtual Mesin pada virtualbox

4.1.2 Konfigurasi Virtual Mesin

Tahapan yang dilakukan adalah :

1. Set *ip address* untuk master dan *node server*

Proses awal setelah membuat dan menginstal virtual mesin adalah *setting ip address.* Adapaun untuk perintah seting ip address yaitu :

nano /etc/network/interfaces)

maka setelah itu akan muncul tampilan seperti pada gambar 4.2

```
  GNU nano 4.8
# interfaces(5) file used by ifup(8) and ifdown(8)
# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

# The loopback network interface
auto lo
iface lo inet loopback

auto enp0s3                             # Nama Interfaces
iface enp0s3 inet dhcp

auto enp0s8                               # Nama Interfaces
iface enp0s8 inet static                  # Mode Static
         address 10.10.10.2/24            # IP Address
```

Gambar 4. 2 konfigurasi *ip address static* dan *dinamis*

Setting semua *ip address server* dengan menggunakan perintah diatas 30ninst address yang dikonfigurasi dengan perintah diatas. Lakukan Hal yang sama pada *Node server.*

2. Instalasi *SSH*

Setelah *ip address* di*setting* yang dilakukan selanjutnya pada virtual mesin adalah menginstall *service* SSH (*Secure Shell*) pada masing-masing *server*. *Secure Shell* berfungsi untuk *server* bisa di*remote* dari jarak jauh.

3. **Pengaturan *password root***

Kemudian atur password sebagai root agar hak akses bisa full ketika melakukan otomasisasi pada *node server* dengan perintah seperti berikut:

passwd root

Perintah ini untuk memberikan password pada user root yang memiliki akses full ke dalam *server*.

4. Generate kode autentikasi untuk *node server* (key ssh keygen)

Adapun perintah untuk generate *SSH key* yaitu:

ssh-keygen atau ssh-keygen t rsa

ssh-keygen berfungsi untuk membuat kode authentikasi pada saat kita *remote* jarak jauh menggunakan protocol ssh. Untuk s**sh-keygen t rsa** perintah yang dipakai untuk generate ssh-keygen dengan kriptografi RSA.

Lakukan langkah ini ke semua *server* baik itu master ataupun *node server.*

Kemudian setelah keygen berhasil didapatkan, copy semuanya keygen *node* ke master dengan perintah:

ssh-copy-id root@ip address *node*

contoh : ssh-copy-id root@10.10.10.3

5. instalasi *Flask*

Kemudian install *Flask* sebagai bahasa permrograman yang akan di[akai pada web. Install *Flask* dengan perintah sebagai berikut:

sudo apt update && sudo apt upgrade

perintah ini berfungsi untuk mengupdate semua packages Linux ke latest version

sudo apt install wget build-essential libncursesw5-dev libssl-dev \ libsqlite3-dev tk-dev libgdbm-dev libc6-dev libbz2-dev libffi-dev zlib1g-dev

perintah ini berfungsi untuk menginstall semua package required

sudo add-apt-repository ppa:deadsnakes/ppa

perintah ini akan menambahkan 32ninstall32 PPA (Personal Package Archive) bernama "deadsnakes/ppa" ke dalam sistem manajemen paket APT (Advanced Package *tool*) di distribusi Linux berbasis Ubuntu atau turunannya.

Apt install *Python*

```
root@master:~# apt install python
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'python-is-python2' instead of 'python'
The following additional packages will be installed:
  libpython2-stdlib libpython2.7-minimal libpython2.7-stdlib python2 python2-minimal python2.7 python2.7-minimal
Suggested packages:
  python2-doc python-tk python2.7-doc binutils binfmt-support
The following NEW packages will be installed:
  libpython2-stdlib libpython2.7-minimal libpython2.7-stdlib python-is-python2 python2 python2-minimal python2.7 python2.7-minimal
0 upgraded, 8 newly installed, 0 to remove and 42 not upgraded.
Need to get 3,815 kB of archives.
After this operation, 16.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Gambar 4. 3 Instal *Flask*

6. Instalasi *Ansible*

kemudian install *Ansible* dengan perintah sebagai berikut:

apt install *Ansible*

pip install *Ansible*

Jika *Ansible* sudah terinstall pada *server* maka hasilnya seperti pada gambar 4.4

```
root@master:~# apt install ansible
Reading package lists... Done
Building dependency tree
Reading state information... Done
ansible is already the newest version (5.10.0-1ppa~focal).
0 upgraded, 0 newly installed, 0 to remove and 42 not upgraded.
root@master:~#
```

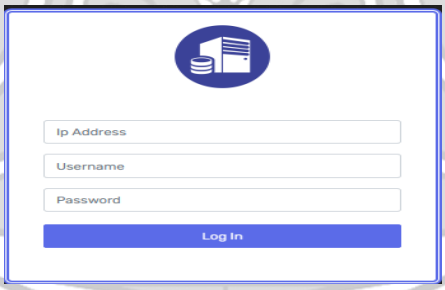
Gambar 4. 4 Install *Ansible*

Pada gambar 4.3 *Ansible* sudah terinstall pada *server* tersebut. *Ansible* berfungsi sebagai alat otomatisasi *open-source* yang digunakan untuk mengotomatisasi tugas-tugas konfigurasi, manajemen, dan penyebaran perangkat lunak. Ini merupakan bagian dari keluarga alat-alat DevOps yang dirancang untuk membantu para *Back End Engineer* IT dalam mengelola infrastruktur dengan cara yang efisien, konsisten, dan dapat diulang.

4.1.3 Desain UI (*User Interface*)

Hasil dari dari desain UI bertujuan untuk melakukan demonstrasi pengaksesan web. Desain ini memiliki beberapa halaman untuk memasukkan atau menampilkan informasi tentang master dan *node server* yang akan diotomatisasi. Berikut desain UI yang dibuat untuk aplikasi web ini.

Halaman *Login*



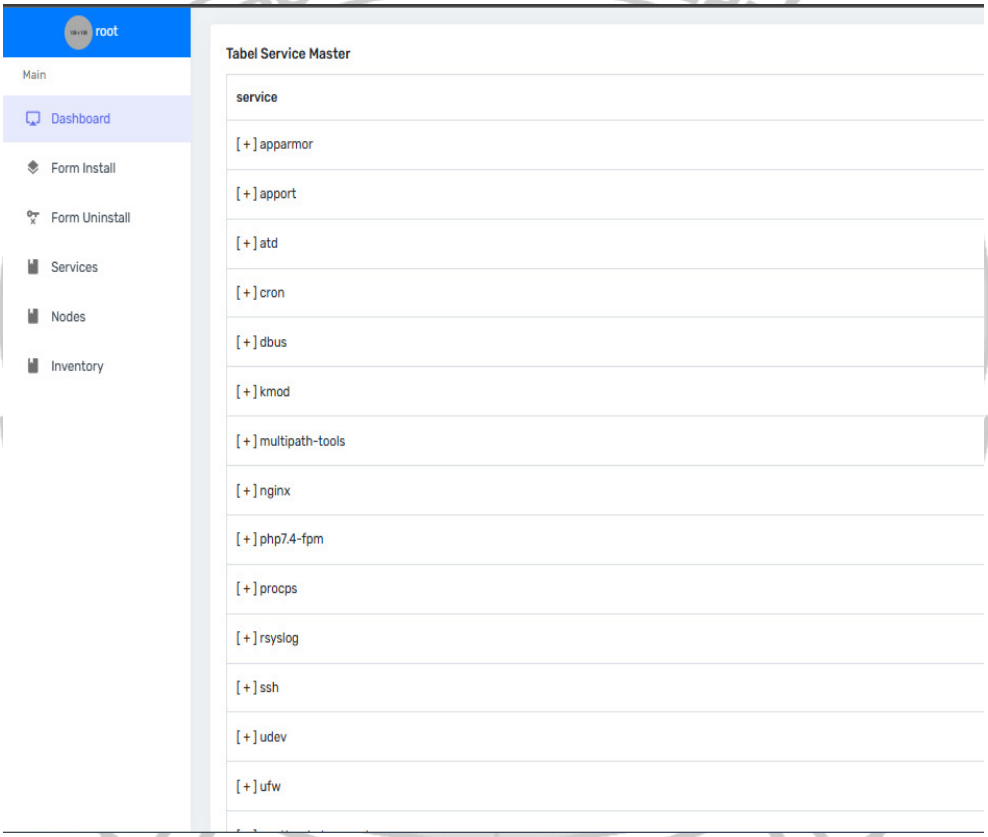
Gambar 4. 5 Tampilan Login

Pada gambar 4.5 adalah gambar halaman yang pertama ketika mengakses web tersebut. Halaman ini merupakan halaman login untuk authentikasi beberapa inputan seperti ip *address*, *username* dan juga *password*. Pada halaman ini juga inputan itu akan diproses ke master *server*, agar *service* pada *master server* bisa

terlihat.

Halaman Login juga akan mengauthentikasi apakah semua kolom yang sudah diisi sudah benar atau tidak. Selain authentikasi itu halaman ini akan mnegirimkan notifikasi apabila master *server* sedang dalam keadaan mati.

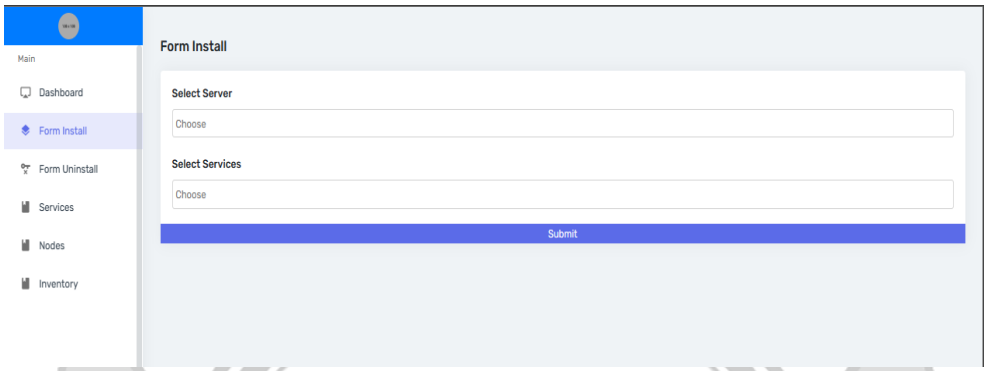
Halaman *Dashboard*



Gambar 4. 6 Tampilan *Dashboard*

Gambar 4.6 menampilkan *Dashboard* ini berfungsi memperlihatkan *service-service* yang terinstall pada master *server*. Tampilan ini juga akan menjadi halaman utama ketika sudah login. *Service-service* yang ditampilkan adalah *service* yang aktif dan berjalan pada master *server*.

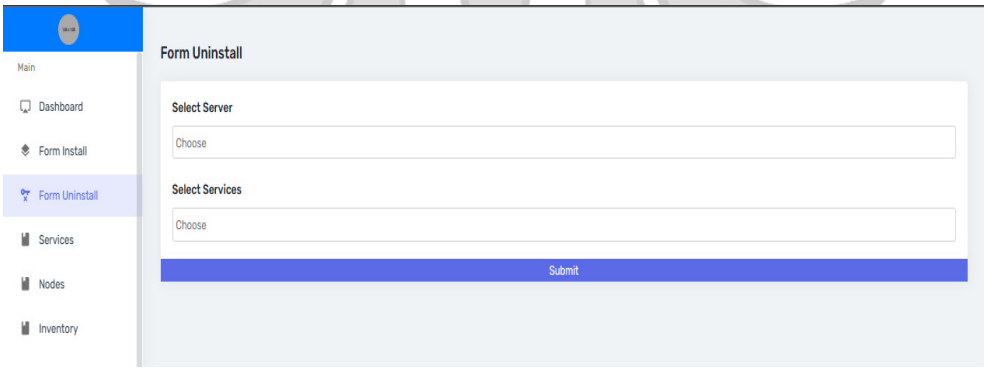
Halaman *Form Install*



Gambar 4. 7 Halaman Form Install

Gambar 4.7 menampilkan halaman form install untuk memilih *node* dan *services* yang akan diinstall disetiap *node*nya. *Node* yang dipilih akan menjadi tujuan akan diinstallnya *service* yang sudah dipilih. Pada form isian bisa memilih secara multiple pada *server* dan *service* form nya.

Halaman Form *Uninstall*

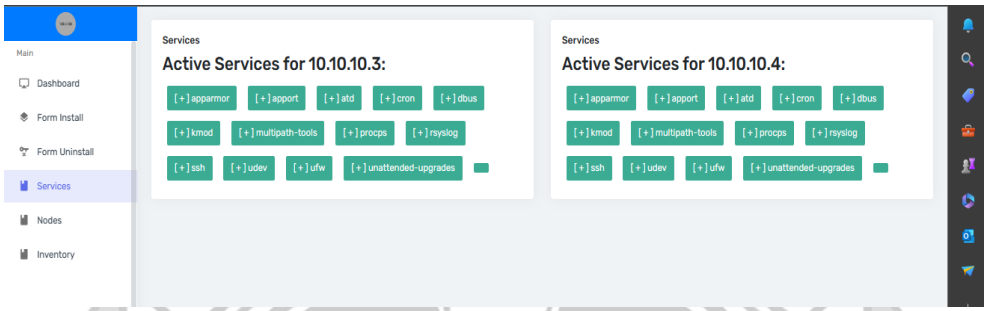


Gambar 4. 8 Halaman Form *Uninstall*

Gambar 4.8 merupakan kebalikan dari gambar 4.7 yaitu *uninstall*. *Uninstall* berfungsi untuk menghilangkan *service* yang ada pada *node server* agar tidak dapat

lagi diakses oleh *user*.

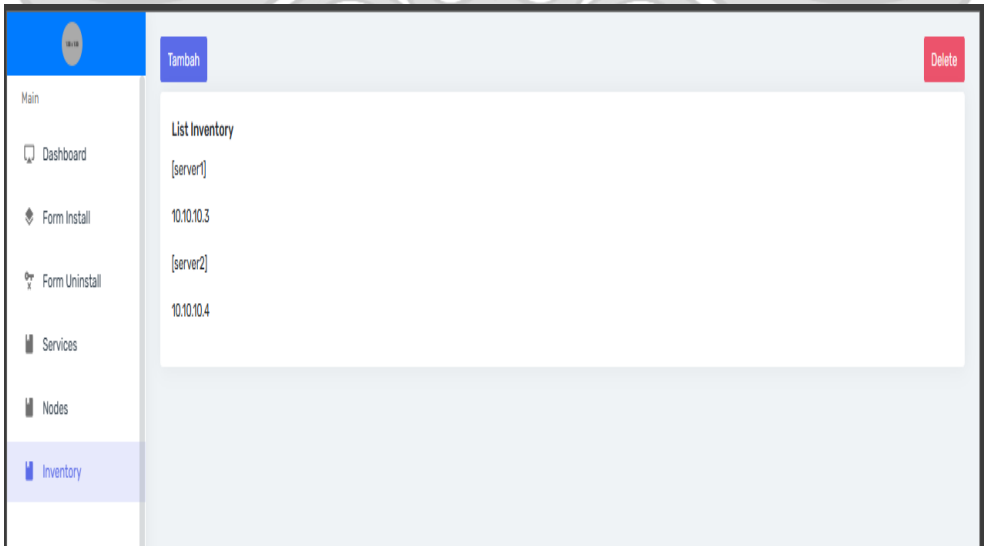
Halaman *Services*

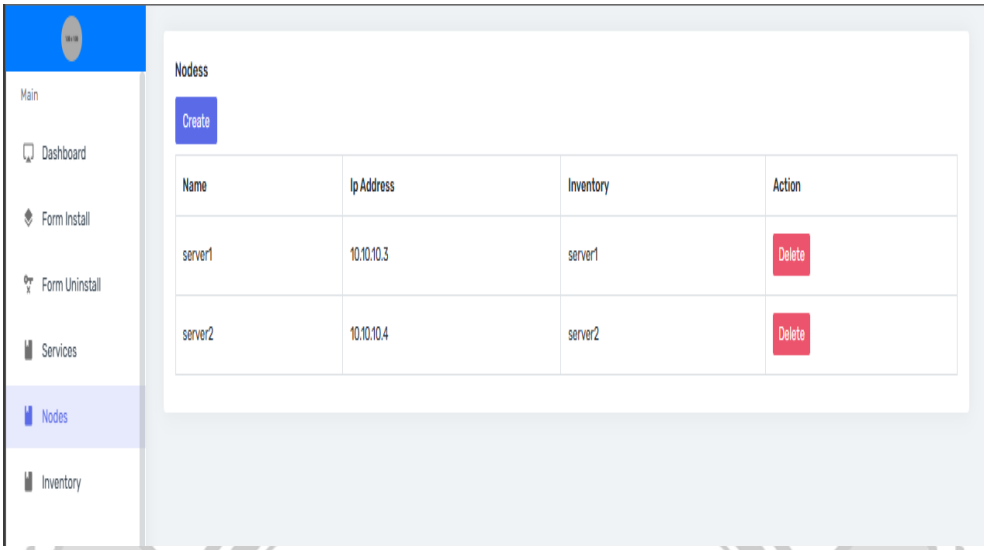


Gambar 4. 9 Halaman *Services*

Gambar 4.9 Halaman *services* ini akan menampilkan *service* pada kedua *node server* seperti *service* Apache, Nginx dan *service-service* lainnya. Halaman *service* ini juga akan menjadi *redirect* ketika *install* dan Uninstall success.

Halaman *List Inventory*





Gambar 4. 10 Halaman List *Inventory* dan *Node*s

Pada gambar 4.10 akan tampil list *Inventory* atau *node server* yang sudah di tambahkan pada file *inventory Ansible*. Agar nantinya *Ansible* bisa menyeleksi semua *server* yang terdaftar pada *inventory*.

4.2 Konfigurasi *Ansible* pada Master *Server*

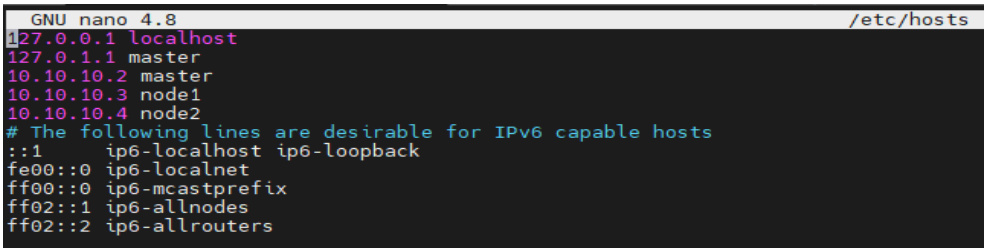
Konfigurasi pada *Ansible* pada master *server* adalah yaitu membuat code yaml dan *Inventory* dari *node server*.

4.2.1 Konfigurasi Host pada setiap *server*

Konfigurasi host pada serve bisa dilakukan dengan perintah sebagai berikut:

nano /etc/hosts

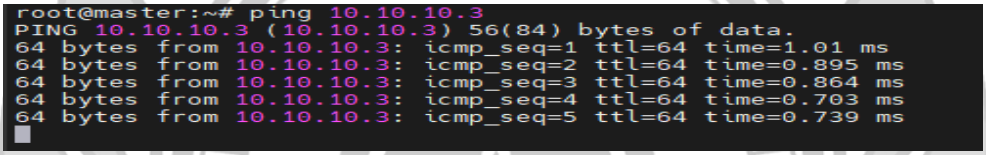
konfigurasi ini berfungsi untuk menghubungkan host antar serve



Gambar 4. 11 *File Host*

Gambar 4.11 menampilkan beberapa nama *host* yang sudah di daftarkan pada master *server*.

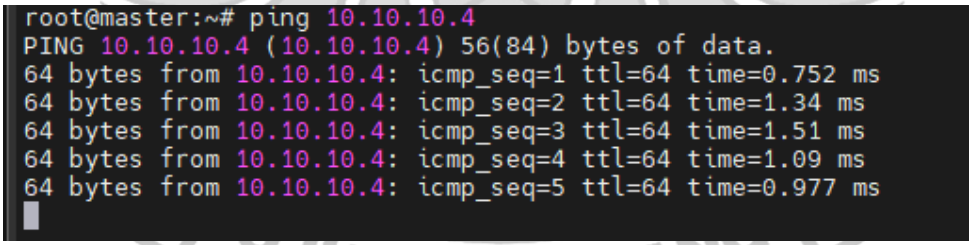
Cek ping antara master dengan *node*1



Gambar 4. 12 Hasil Ping Ke *node* 1

Gambar 4.12 memperlihatkan hasil ping dari master ke *node server*. Hasil yang tampil adalah berhasil terhubung ke *node*1.

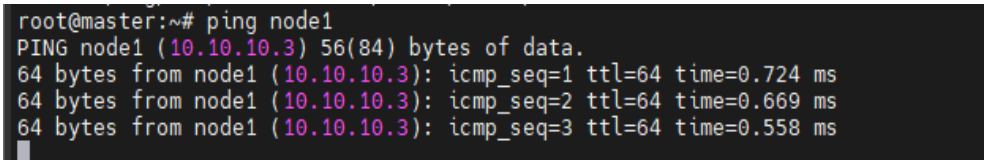
Cek ping antara master dengan *node*2



Gambar 4. 13 hasil Ping Ke *node* 2

Gambar 4.13 memperlihatkan hasil ping dari master ke *node server*. Hasil yang tampil adalah berhasil terhubung ke *node*2.

Cek ping juga bisa dilakukan dengan *hostname*



Gambar 4. 14 hasil Ping Memakai *Hostname*

Gambar 4.14 menampilkan hasil dari ping menggunakan *hostname*

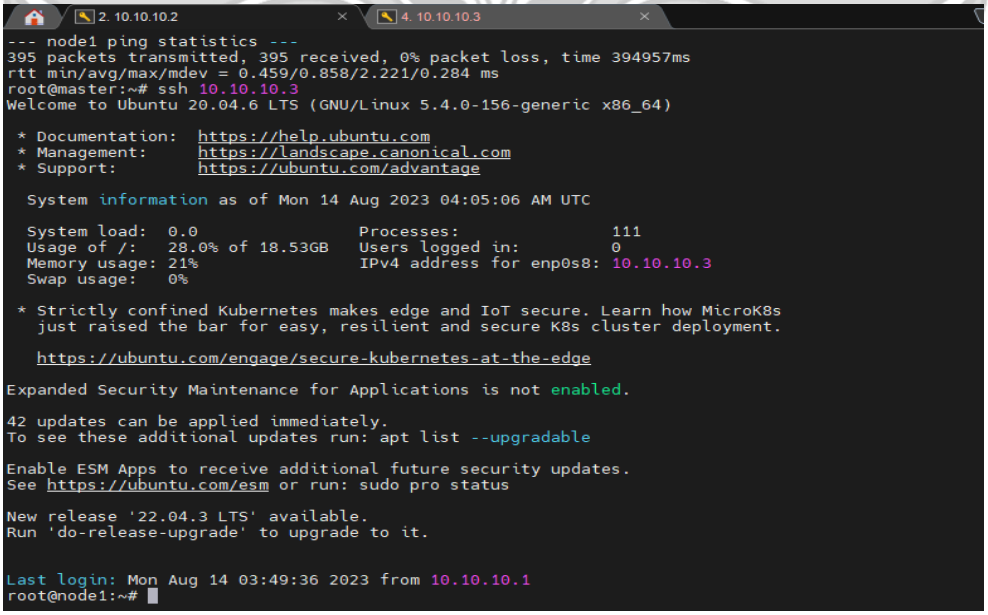
4.2.2 Tes Remot Tanpa password

Tes remot tanpa password bertujuan agar *Ansible* dapat menggunakan *node server* tanpa authentikasi password terlebih dahulu.

Tes *remote* ke *node* 1 dengan perintah sebagai berikut:

shh ip_*node* atau ssh 10.10.10.3

tampilan yang akan muncul jika berhasil adalah seperti pada gambar 4.15



Gambar 4. 15 Hasil Sukses Remot *node*1

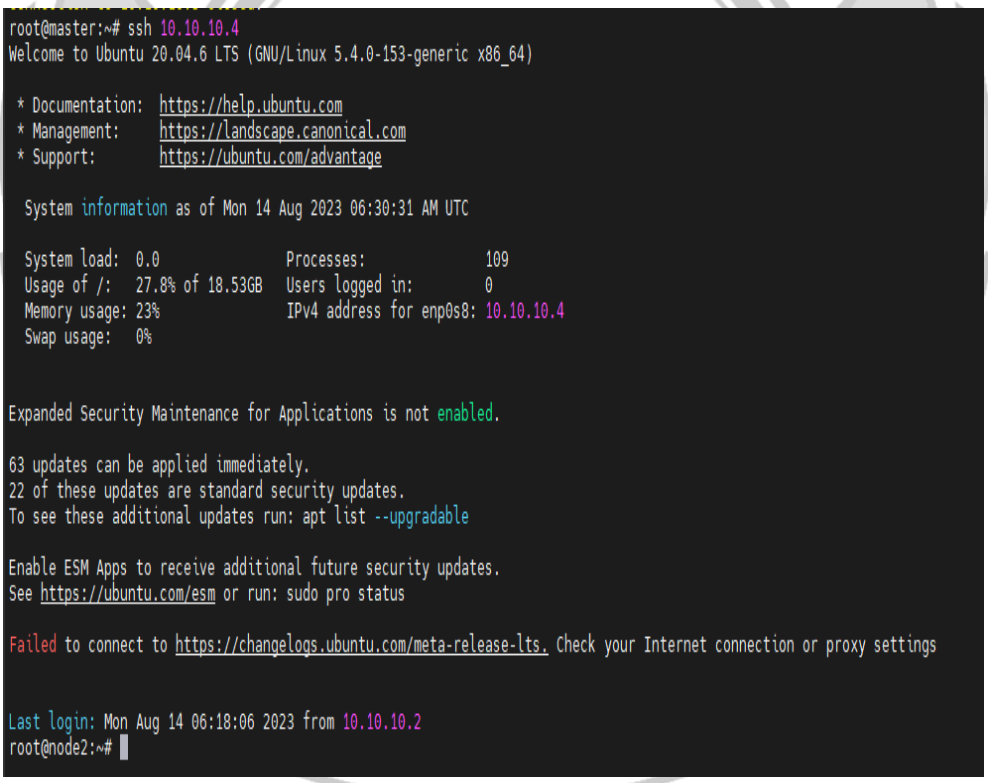
Gambar 4.15 Menampilkan hasil ketika remot ke *node* 1 yang berhasil dan

pengaturan SSH yang sudah benar. Dan *node*1 siap untuk dieksekusi dengan *Ansible.*

Tes Remot ke *node*2

Tes remot ke *node*2 bertujuan untuk melihat apakah konfigurasi SSH berhasil atau tidak. Perintah tes seperti pada tes remot ke *node* 1 dengan ip yang berbeda, berikut perintah untuk tes *remote* ke *node*2 sebagai berkut:

ssh 10.10.10.4



Gambar 4. 16 Hasil *Remote node* 2

Gambar 4.16 Menampilkan hasil *remote* ke *node* 2 yang berhasil tanpa password.

4.2.3 Membuat Program *Ansible*

Program *Ansible* ini berfungsi untuk mengeksekusi semua command Linux yang telah dibuat dalam satu file yaml. Program *Ansible* ini akan dieksekusi melalui web dengan *framework Flask* dengan *library* Paramiko.

Program *Ansible* ini selain akan jadi pengesekusi. *Ansible* juga akan memberikan pengaman yang akan melindungi komunikasi antar *node*.

Program *Ansible* untuk menginstal *service nginx* dan *library* pendukungnya seperti pada gambar 4.17

```
  GNU nano 4.8
- name: Install Nginx and PHP-FPM
  hosts: server1
  become: yes

  tasks:
    - name: ensure nginx is at the latest version
      apt:
        name: nginx
        state: latest

    - name: start nginx
      service:
        name: nginx
        state: started

    - name: ensure php-fpm is installed
      apt:
        name: php-fpm
        state: present

    - name: start php-fpm
      service:
        name: php-fpm
        state: started
```

Gambar 4. 17 Program Nginx *Ansible*

Pada Gambar 4.17 merupakan program untuk menginstal nginx melalui *Ansible* . Program ini akan menginstal nginx beserta *package library* dari nginxnya sehingga ketika programnya dieksekusi nginx akan terinstal dengan *package nginx*.

Program ini akan menginstal dan juga menjalankan nginx dan *library* yang sudah dicantumkan pada program. Program ini akan dieksekusi pada *node*1 karena hosts yang dituju adalah *server*1. Versi yang nginx yang akan terinstal juga adalah versi yang terbaru.

Program *Ansible* untuk menguninstal *service* nginx dan juga *library* pendukungnya seperti pada gambar 4.18

```
  GNU nano 4.8                                                uninstal
- name: Uninstall Nginx and PHP
  hosts: server1
  become: true

  tasks:
    - name: stop nginx
      service:
          name: nginx
          state: stopped
    - name: ensure nginx is not installed
      apt: name=nginx state=absent
```

Gambar 4. 18 Program *Uninstall Nginx*

Pada gambar 4.18 merupakan program untuk *uninstall nginx* dan *library* pendukung dari nginx.

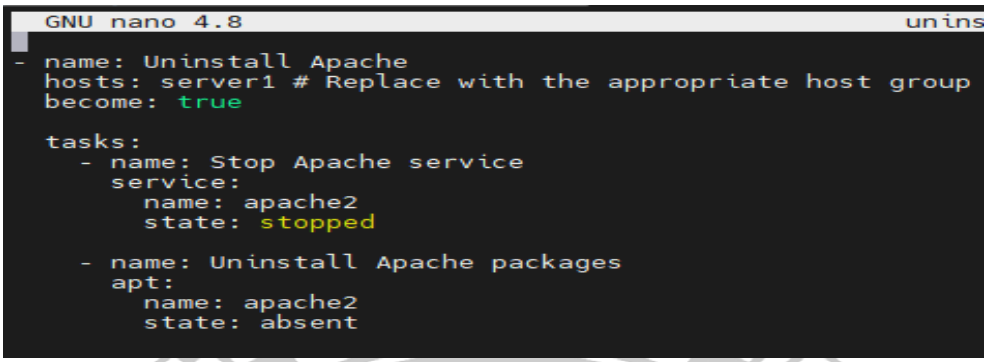
Program *Ansible* untuk menginstal apache2 dan juga *library* pendukung apache2 dapat dilihat pada gambar 4.19

```
  GNU nano 4.8
- name: Install Apache
  hosts: server1  # Ganti dengan grup host yang sesuai
  become: true

  tasks:
    - name: Install Apache
      apt:
        name: apache2
        state: present
```

Gambar 4. 19 Program *Install* Apache2 *Ansible*

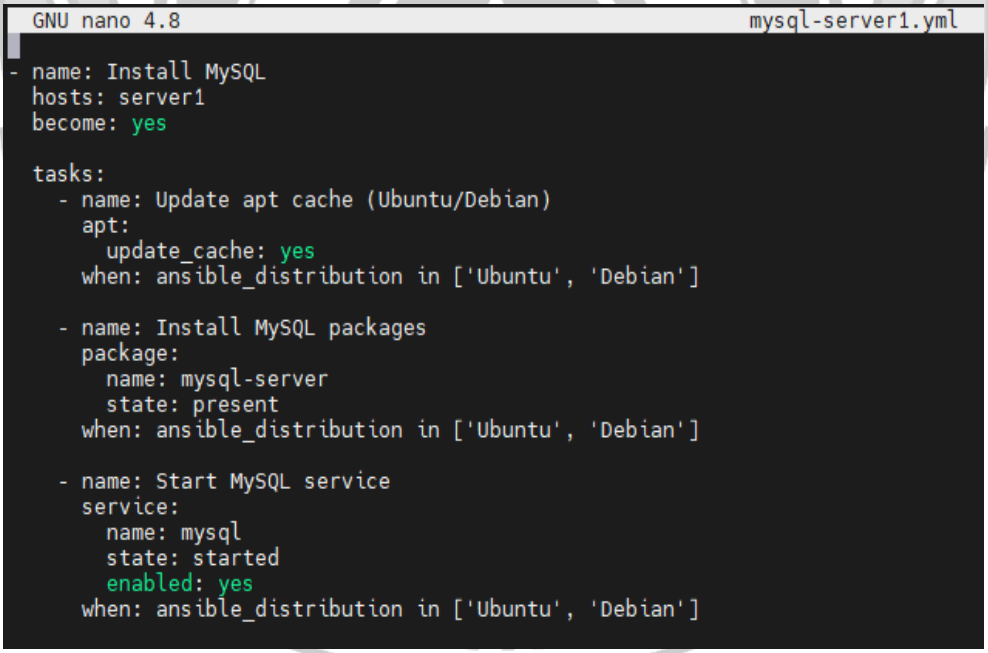
Pada gambar 4.19 merupakan program untuk menginstall *service* apache2 dengan *library* pendukungnya.

Program untuk menguninstal apache2 seperti pada gambar 4.20



Gambar 4. 20 Program *Uninstall* Apache2 *Ansible*

Pada gambar 4.20 menunjukkan program *Ansible* untuk menguninstal Apache. Program *Ansible* untuk instal mysql dapat dilihat pada gambar 4.21



Gambar 4. 21 Program *Install Mysql*

Pada gambar 4.21 terlihat program untuk menginstal dan mengaktifkan *service* mysql dengan *Ansible.*

Program *uninstal Mysql* dapat dilihat pada gambar 4.22

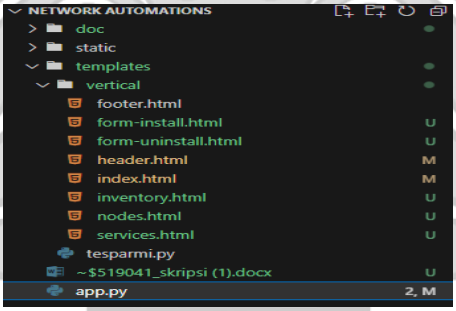
```
  GNU nano 4.8                                          uninstall-mysq
- name: Uninstall MySQL
  hosts: server1
  become: yes

  tasks:
    - name: Stop MySQL service
      service:
        name: mysql
        state: stopped
      when: ansible_distribution in ['Ubuntu', 'Debian']

    - name: Uninstall MySQL packages
      package:
        name: mysql-server
        state: absent
      when: ansible_distribution in ['Ubuntu', 'Debian']

    - name: Delete MySQL data directory
      file:
        path: /var/lib/mysql
        state: absent
      when: ansible_distribution in ['Ubuntu', 'Debian']
```

Gambar 4. 22 Program *Uninstall Mysql*

Pada gambar 4.22 terlihat program untuk menghentikan dan menguninstal *service* Mysql pada *node server*.

4.3 Implementasi Web *Server*

Proses membuat program web dengan struktur pada gambar 4.23



Gambar 4. 23 Struktur Program Web

Pada gambar 4.23 terlihat struktur file dari web aplikasi otomasi. Pada folder templates adalah template setiap halaman web yang dipakai pada aplikasi. Pada *folder static* berisi gambar yang akan dipanggil ataupun material-material yang akan mengisi halaman web. Dan terakhir ada file app.py yang berisi program

backend yang menjalankan web.

Untuk penjelasan kode proses aplikasi pada file app.py akan dijelaskan sebagai berikut.

a. Proses *Login*

Proses *login* dapat diliat pada gambar 4.24

```
@app.route("/", methods=['GET', 'POST'])
def index():
    global ip_add, cookie, node_name, username, password
    if request.method == 'GET':
        return render_template('vertical/pages-login.html')

    if request.method == 'POST':
        ip_add = request.form['ip_address']
        username = request.form['username']
        password = request.form['password']
        session['username'] = username
        session['password'] = password
        session['ipadd'] = ip_add

        query = "SELECT * FROM users WHERE username = %s AND password = %s"
        mycursor.execute(query, (username, password))
        # Periksa hasil query
        result = mycursor.fetchone()
        if result:
            user_id = result[0]
            session['id'] = user_id
        else:
            flash('Username atau password salah', 'error')
            return render_template('vertical/pages-login.html')
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        if check_server_status(ip_add, username, password):
            ssh = paramiko.SSHClient()
            ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
            ssh.connect(hostname=ip_add, username=username, password=password)
            cmd = "service --status-all | grep +"
            # Jalankan perintah pada host target
            stdin, stdout, stderr = ssh.exec_command(cmd)
            output = stdout.read().decode()
            daftar_layanan = output.split('\n')
            ssh.close()
            return redirect(url_for('dashboard'))
        else:
            flash('Server Masih Dalam Keadaan Mati', 'error')
            return render_template('vertical/pages-login.html')
```

Gambar 4. 24 Funtion Proses *Login*

Pada gambar 4.24 dijelaskan proses *login* pada *function index*. Untuk metode *GET* digunakan untuk menampilkan halaman *login*. Untuk metode *POST* dipakai untuk memasukkan inputan dari halaman *login*, dicek pada *database* untuk memastikan data ada atau tidak ada. Kemudian data akan di pakai untuk *login* ke Master *server* menggunakan *library* Paramiko dan *service SSH*. Pada proses ini ssh dan Paramiko akan mengecek apakah *server* dalam keadaan hidup atau mati. Apabila dalam keadaan hidup maka akan diarahkan ke *Dashboard*. Sedangkan jika *server* dalam keadaan mati maka kembali ke halaman login dengan *alert error*.

b. *Dashboard*

Kode *Dashboard* dapat diliat pada gambar 4.25

```python
@app.route("/dashboard", methods=['GET'])
def dashboard():
        username = session.get('username')
        password = session.get('password')
        ip_add = session.get('ipadd')
        if not username:
            return redirect(url_for('index'))

        # print(username)

        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        ssh.connect(hostname=ip_add, username=username, password=password)
        cmd = "service --status-all | grep +"
        # Jalankan perintah pada host target
        stdin, stdout, stderr = ssh.exec_command(cmd)
        output = stdout.read().decode()
        daftar_layanan = output.split('\n')
        ssh.close();
        return render_template('vertical/index.html',username=username, service=daftar_layanan)
```

Gambar 4. 25 Kode Untuk Menampilkan Halaman *Dashboard*

Pada gambar 4.25 terlihat kode untuk menampilkan *service* pada master *server*.

c. Proses *Install*

Kode proses *install* dapat diliat pada gambar 4.26

```python
@app.route("/form-install", methods=['GET', 'POST'])
def formInstall():
    username = session.get('username')
    password = session.get('password')
    ip_add = session.get('ipadd')
    master = session.get('id')
    if not username:
        return redirect(url_for('index'))
    if request.method == 'GET':
        query = "SELECT * FROM nodes Where master=%s "
        query2 = "SELECT * FROM services where status=1"
        mycursor.execute(query,(master,))
        servers = mycursor.fetchall()  # Fetch all rows returned by the query
        mycursor.execute(query2)
        services = mycursor.fetchall()  # Fetch all rows returned by the query
        return render_template('vertical/form-install.html', servers=servers, services=services)

    if request.method == 'POST':
        selected_servers = request.form.getlist('selected_servers[]')
        selected_services = request.form.getlist('selected_services[]')
        if not selected_servers and selected_services:
            flash("Server atau service belum terisi", "error")
            return redirect(url_for('formInstall'))
        if not selected_servers:
            flash("Server atau service belum terisi", "error")
            return redirect(url_for('formInstall'))
        if not  selected_services:
            flash("Server atau service belum terisi", "error")
            return redirect(url_for('formInstall'))

        try:
            ssh = paramiko.SSHClient()
            ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
            ssh.connect(hostname=ip_add, username=username, password=password)
            # command = "cd /home/muhtaufikw/"
            # stdin, stdout, stderr = ssh.exec_command(command)
            for server in selected_servers:
                for service in selected_services:
                    cmd = f"ansible-playbook -i inventory {service}-{server}.yml"
                    # cmd = "pwd"
                    # print(cmd)
                    # Jalankan perintah pada host target
                    stdin, stdout, stderr = ssh.exec_command(cmd)
                    output = stdout.read().decode()
```

Gambar 4. 26 Kode Proses Instal

Pada gambar 4.26 adalah function untuk menginstal *service*. Untuk metode *GET* berfungsi untuk menampilkan menu *form* instal. Sedangankan untuk metode POST digunakan untuk memproses masukan yang ada pada inputan pada halaman *form* yang akan langsung dieksekusi ke master dengan *SSH* dan Paramiko, lalu dilanjutkan dengan eksekusi *node server* dengan *Ansible.*

d. Proses *Uninstall*

Kode proses *install* dapat diliat pada gambar 4.27

```
@app.route("/form-uninstall", methods=['GET', 'POST'])
def formUninstall():
    username = session.get('username')
    password = session.get('password')
    ip_add = session.get('ipadd')
    master = session.get('id')
    if not username:
        return redirect(url_for('index'))
    if request.method == 'GET':
        query = "SELECT * FROM nodes where master= %s "
        query2 = "SELECT * FROM services where status=0"
        mycursor.execute(query,(master,))
        servers = mycursor.fetchall()  # Fetch all rows returned by the query
        mycursor.execute(query2)
        services = mycursor.fetchall()  # Fetch all rows returned by the query
        return render_template('vertical/form-uninstall.html', servers=servers, services=services)

    if request.method == 'POST':
        selected_servers = request.form.getlist('selected_servers[]')
        selected_services = request.form.getlist('selected_services[]')
        if not selected_servers and selected_services:
            flash("Server atau service belum terisi", "error")
            return redirect(url_for('formUninstall'))
        if not selected_servers:
            flash("Server atau service belum terisi", "error")
            return redirect(url_for('formUninstall'))
        if not  selected_services:
            flash("Server atau service belum terisi", "error")
            return redirect(url_for('formUninstall'))
        try:
            ssh = paramiko.SSHClient()
            ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
            ssh.connect(hostname=ip_add, username=username, password=password)
            # command = "cd /home/muhtaufikw/"
            # stdin, stdout, stderr = ssh.exec_command(command)
            for server in selected_servers:
                for service in selected_services:
                    cmd = f"ansible-playbook -i inventory uninstall-{service}-{server}.yml"
                    # cmd = "pwd"
                    # print(cmd)
                    # Jalankan perintah pada host target
                    stdin, stdout, stderr = ssh.exec_command(cmd)
                    output = stdout.read().decode()
                    errors = stderr.read().decode()
```

Gambar 4. 27 Proses *Uninstall*

Pada gambar 4.27 terlihat proses *uninstall* yang mirip dengan *install* cuman beda proses.

e. Tampilan *service*

Kode untuk menampilkan *service* dapat dilihat pada gambar 4.28

```
@app.route("/services", methods=['GET'])
def services():
    username = session.get('username')
    # password = session.get('password')
    # ip_add = session.get('ipadd')
    master = session.get('id')
    if not username:
        return redirect(url_for('index'))


    mycursor.execute('SELECT username, ip_address, password FROM nodes where master =%s',(master,))
    servers_info = mycursor.fetchall()
    print(servers_info)
    active_ports_by_server = {}

    for server_info in servers_info:
        username, ip_add, password = server_info

        # Connect to the remote host using SSH
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

        try:
            if check_server_status(ip_add, username, password):
                ssh = paramiko.SSHClient()
                ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
                ssh.connect(hostname=ip_add, username=username, password=password)
                cmd = "service --status-all | grep +"
                # Jalankan perintah pada host target
                stdin, stdout, stderr = ssh.exec_command(cmd)
                output = stdout.read().decode()
                daftar_port = output.split('\n')
                ssh.close()
                active_ports_by_server[ip_add] = daftar_port

            else:
                flash('Node Masih Dalam Keadaan Mati', 'error')
                return redirect(url_for('dashboard'))
        except paramiko.AuthenticationException:
            # Handle authentication errors, if any
            active_ports_by_server[ip_add] = ["Authentication failed"]
        finally:
            ssh.close()
```

Gambar 4. 28 Kode Menampikan *Service*

Pada gambar 4.28 menampilkan kode untuk tampilan *service* dengan data dari *service* kedua *node*.

f. Inventory

Kode *inventory* dapat diliat pda gambar 4.29

```
@app.route('/inventory',  methods=['GET', 'POST'])
def lihat_inventory():
    username = session.get('username')
    password = session.get('password')
    ip_add = session.get('ipadd')
    if not username:
        return redirect(url_for('index'))

    if request.method == 'GET':
        remote_file_path = '/root/inventory'
        inventory_contents = read_inventory_file(ip_add, username, password, remote_file_path)
        lines = inventory_contents.split('\n')
        processed_lines = [line.strip() for line in lines[2:] if line.strip()]
        return render_template('vertical/inventory.html', isi_inventory=processed_lines)
    if request.method == 'POST':
        nama_host = request.form['server']
        alamat_ip = request.form['ip_address']
        if not nama_host or not alamat_ip:
            return "Nama Host dan Alamat IP tidak boleh kosong"
        remote_file_path = '/root/inventory'
        add_new_ip_to_inventory(ip_add, username, password,nama_host, remote_file_path, alamat_ip)
        return redirect(url_for("lihat_inventory"))

@app.route('/delete-inventory',  methods=['POST'])
def delete_inventory():
    username = session.get('username')
    password = session.get('password')
    ip_add = session.get('ipadd')
    num_lines_to_delete = 1
    if request.method == 'POST':
        remote_file_path = '/root/inventory'
        delete_last_lines(ip_add, username, password, remote_file_path, num_lines_to_delete)
        return redirect(url_for("lihat_inventory"))

@app.route("/nodes", methods=['GET', 'POST'])
```

Gambar 4. 29 Kode *Inventory*

Pada gambar 4.29 memperlihatkan kode untuk menampilkan *inventory* dari file *inventory* pada master. Dan juga dapat menghapus isi dua baris file terakhir pada file tersebut. Setelah berhasil akan diarahkan ke halaman *inventory* dengan *alert* sukses.

g. *Node*s

kode *node* bisa diliat pada gambar 4.30

```
@app.route("/nodes", methods=['GET', 'POST'])
def nodes():
    master = session.get('id')
    username = session.get('username')
    if not username:
        return redirect(url_for('index'))
    if request.method == 'GET':
        #    mycursor.execute(query,(master,))
        # servers = mycursor.fetchall()  # Fetch all rows returned by the query
        # mycursor.execute(query2)
        # services = mycursor.fetchall(
        query = "SELECT * FROM nodes Where master= %s "
        query2 = "SELECT * FROM users "
        mycursor.execute(query,(master,))
        servers = mycursor.fetchall()
        mycursor.execute(query2)
        masters = mycursor.fetchall()
        return render_template("vertical/nodes.html", nodes=servers, masters=masters)
    if request.method == 'POST':
        name = request.form.get('name')
        ip_add = request.form.get('ip_add')
        inventory = request.form.get('inventory')
        username = request.form.get('username')
        password = request.form.get('password')
        master_id = request.form.get('master')  # Assuming master_id is the value from the dropdown option

        # Assuming your table columns are 'name', 'ip_add', 'inventory', 'username', 'password', and 'master'
        insert_query = "INSERT INTO nodes (name, ip_address, inventory, username, password, master) VALUES (%s, %s, %s, %s, %s, %s)"
        values = (name, ip_add, inventory, username, password, master_id)
        mycursor.execute(insert_query, values)
        mydb.commit()
        return redirect(url_for('nodes'))

@app.route("/delete-node/<int:id>", methods=['POST'])
def delete_node(id):

    if request.method == 'POST':
        # Assuming you want to delete the node with the provided ID
        delete_query = "DELETE FROM nodes WHERE id = %s"
        mycursor.execute(delete_query, (id,))
        mydb.commit()
        return redirect(url_for('nodes'))
```

Gambar 4. 30 Kode *Node*s

Pada gambar 4.30 terlihat kode untuk menampilkan *node* dengan data dari *database* selanjutnya *node* bisa dibuat ataupun dihapus. Jika *node* telah dibuat, selanjutnya daftarkan *node* tersebut pada halaman *inventory*.

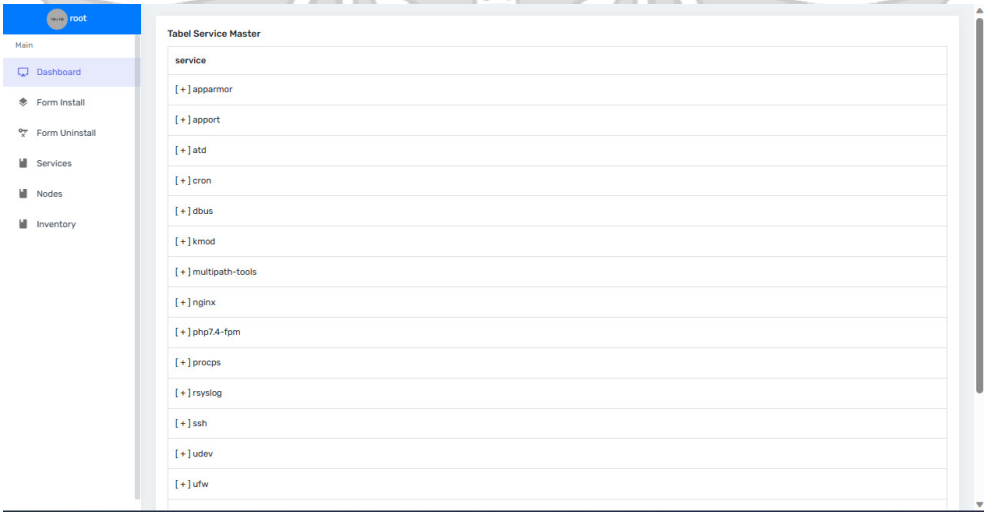
4.4 Proses Penggunaan Aplikasi

Proses Login terlihat pada gambar 4.31

Gambar 4. 31 Halaman *Login*

Gambar 4.31 menujukkan proses login dengan mengisi *Ip address*, *username* dan *password master server*. Isikan *user root* agar user bisa mendapatkan hak akses *full* pada master.

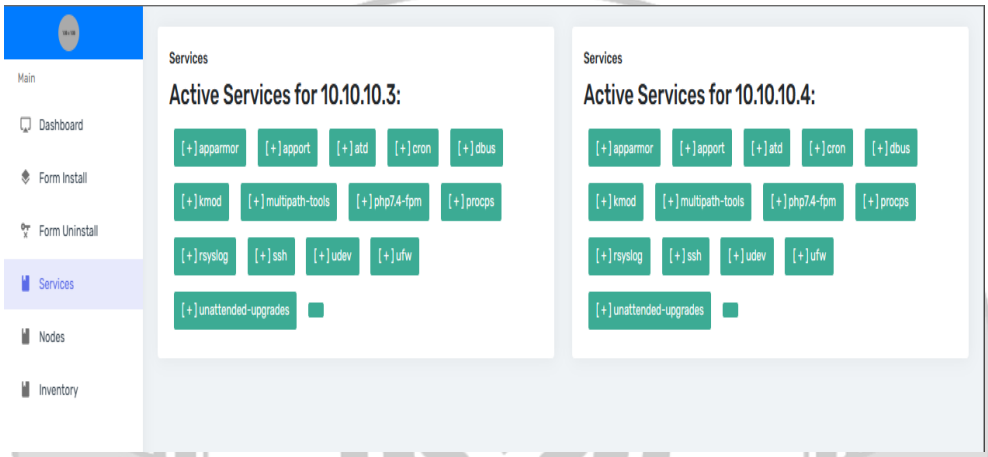
Proses setelah login diperlihatkan pada gambar 4.32



Gambar 4. 32 Halaman *Dashboard*

Gambar 4.32 menujukkan hasil setelah *Ip address*, *username* dan *password* yang benar dan *server* dalam keadaan hidup. Halaman ini juga akan menampilkan *service-service* yang sedanng berjalan pada master *server*.

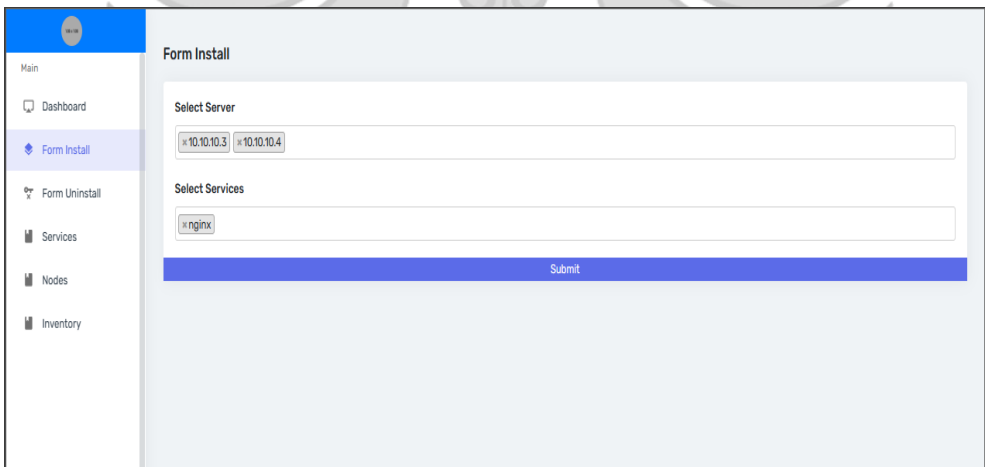
Lihat *service* pada aplikasi terlihat pada aplikasi 4.33



Gambar 4. 33 Halaman *Service*

Pada gambar 4.33 *service* aplikasi yang akan diinstal belum terdapat pada layanan *service*.

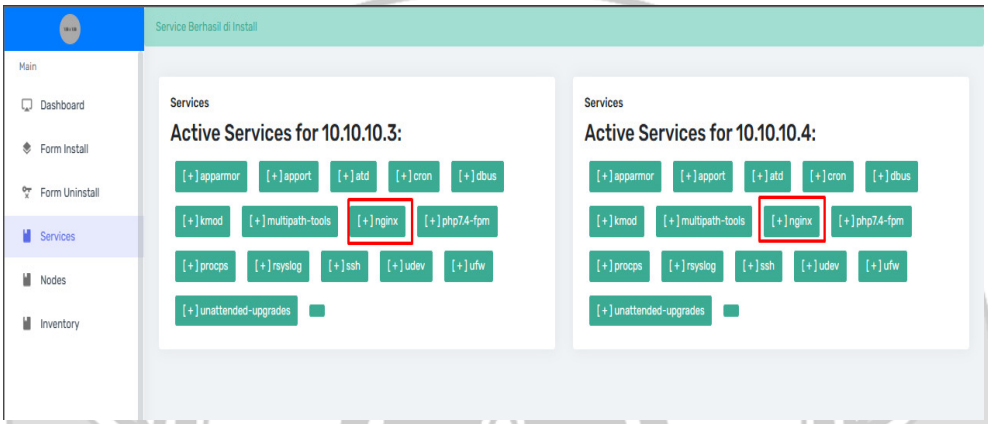
Proses instal aplikasi pada *form install* 4.34



Gambar 4. 34 *Testing* aplikasi pada web

Pada gambar 4.34 *form install* telah diisi untuk instalasi nginx pada kedua *node server*. Pada halaman ini anda bisa memilih *server* dan *service* sesuai dengan kebutuhan *install.*

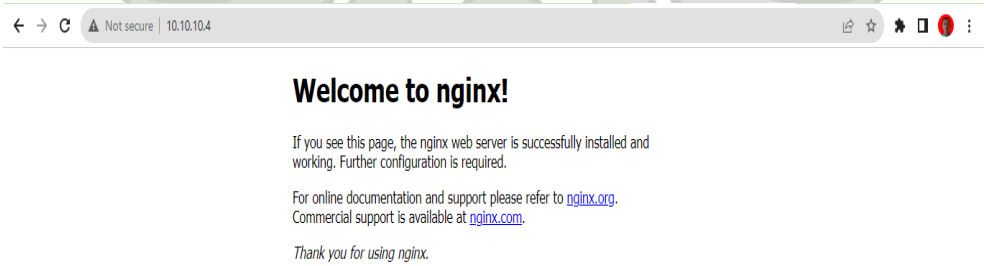
Hasil *service* aplikasi pada bisa dilihat pada gambar 4.35



Gambar 4. 35 Halaman *Service* setelah Install

Gambar 4.35 memperlihatkan *service* sudah terinstall pada kedua *node server*.

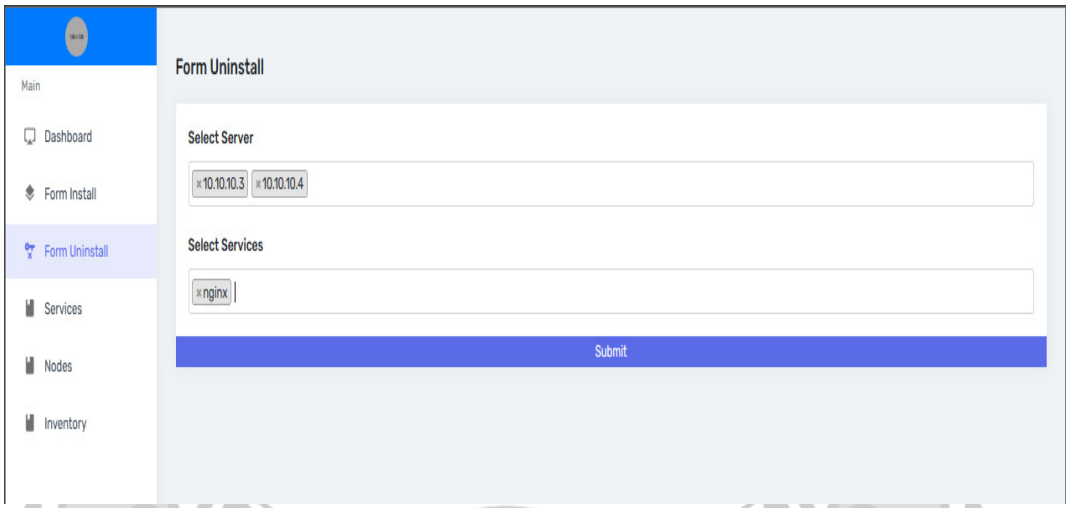
Hasil *testing* aplikasi bisa diliat gambar 4.36



Gambar 4. 36 *Testing Nginx*

Pada gambar 4.36 adalah hasil setelah aplikasi di *install* pada web aplikasi dan hasilnya sudah berhasil karena nginx telah dapat diakses.

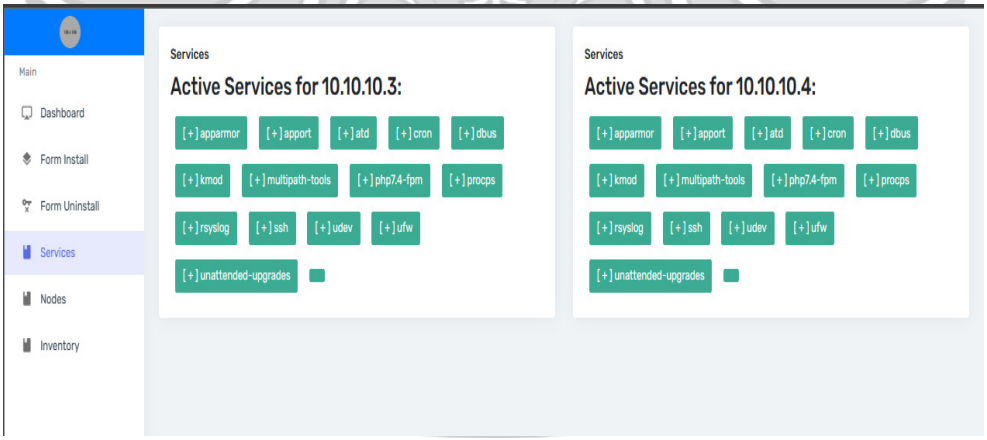
Uninstall service dengan mengisi *form uninstall* seperti pada gambar 4.37



Gambar 4. 37 Halaman *Form Uninstall*

Pada gambar 4.37 mengisi form *uninstall* untuk menguninstal nginx pada kedua *node server*.

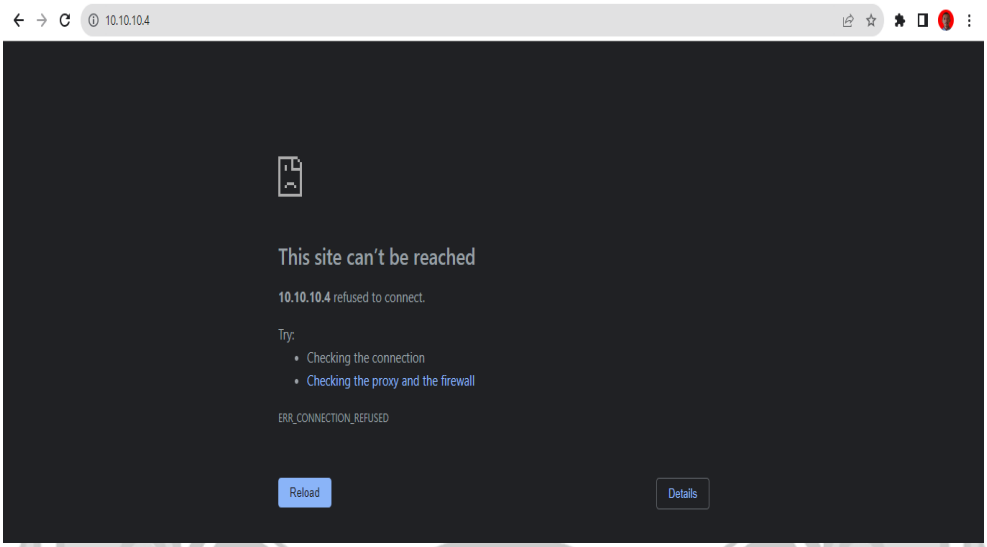
Halaman *services* setelah nginx diuninstal terlihat pada gambar 4.38



Gambar 4. 38 halaman *Service* Setelah *Uninstall*

Pada gambar 4.38 terlihat *service* nginx telah teruninstal pada kedua *node server*.

Hasil *testing* pada web terlihat pada gambar 4.39

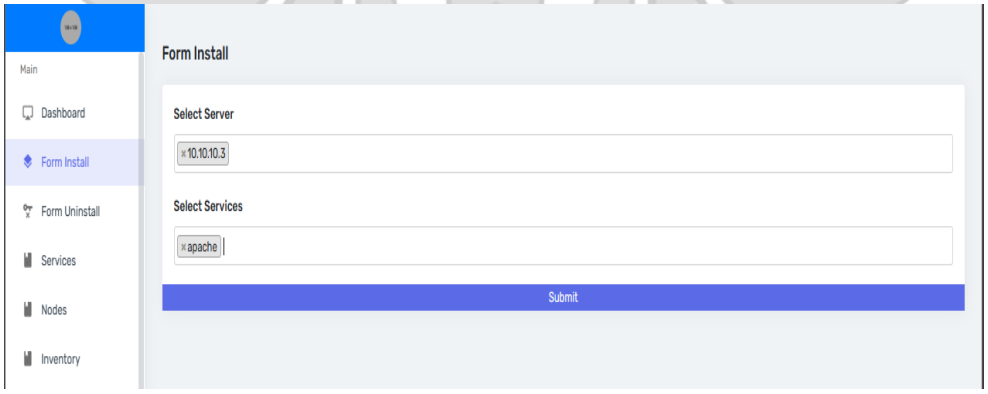


Gambar 4. 39 *Testing Nginx*

Pada gambar 4.39 terlihat *service* telah teruninstal pada *node server*. Artinya *service* yang nginx yang dipilih pada menu *form uninstall* telah dihilangkan.

Selanjutnya testing aplikasi dengan *service* Apache2

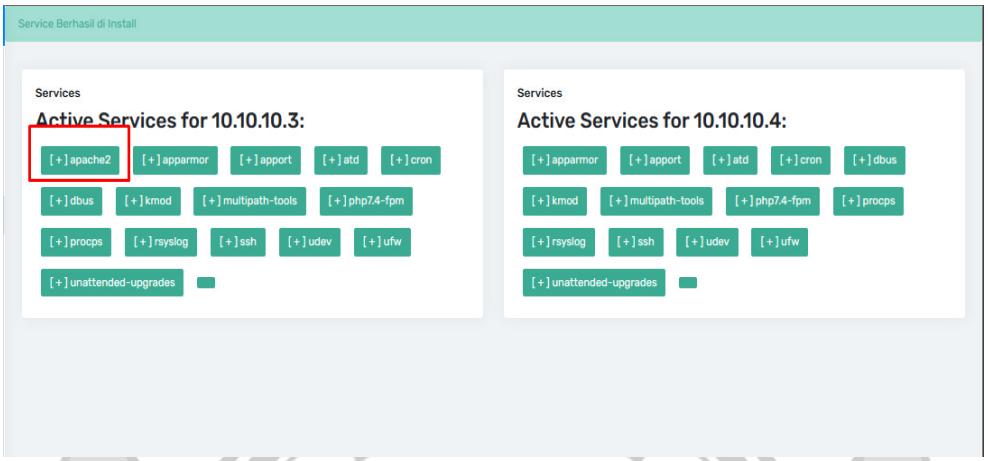
Instal Apache2 dapat dilihat pada gambar 4.40



Gambar 4. 40 *install* Apache2

Pada gambar 4.40 terlihat tampillanya seperti pada saat *install* Nginx dengan memilih *server* dan *service* yang akan diinstal

Ketika berhasil diinstal maka akan muncul halaman seperti pada gambar 4.41



Gambar 4. 41 Hasil Setelah *install* Apache

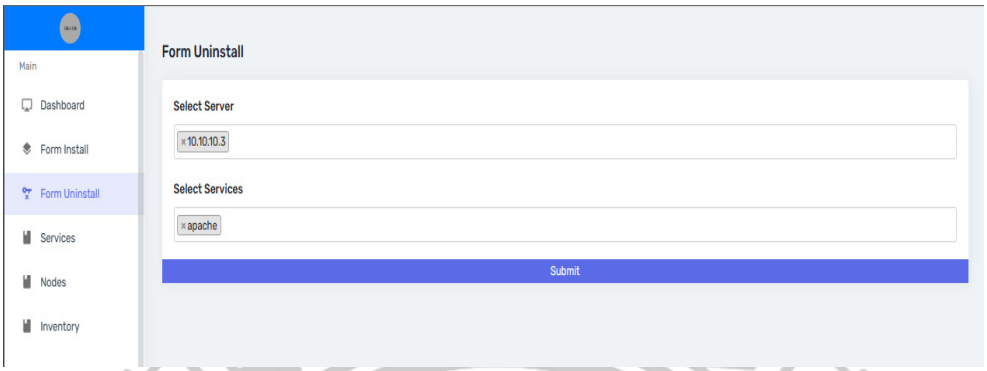
Pada gambar 4.41 terlihat *service* yang diinstal pada *server* 10.10.10.3 atau *node* 1 sudah bertambah Apache2.

Kemudian *Testing* Aplikasi pada halaman web dapat diliat pada gambar 4.42

Gambar 4. 42 Hasil *Testing* Apache2

Pada gambar 4.42 tampil hasil penelusuran ip pada *node* 1 yang sudah terinstal Apache2.

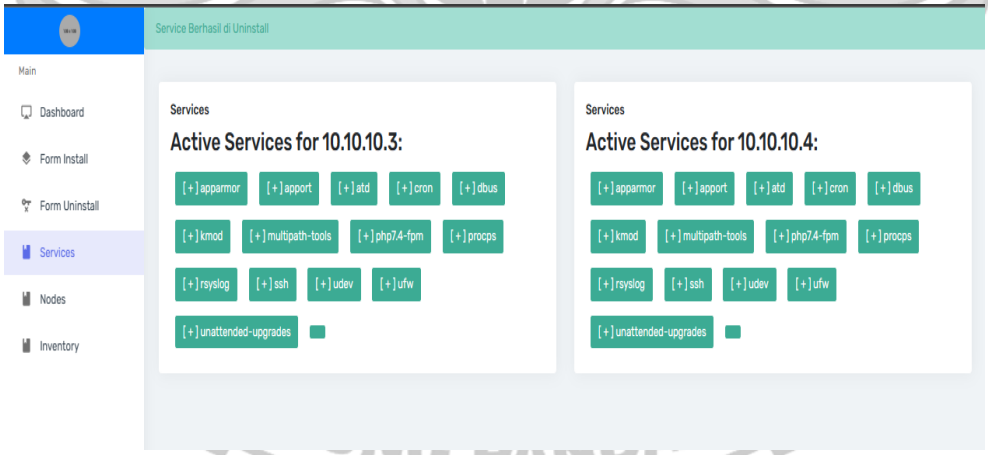
Kemudian *uninstall* Apache2 pada gambar 4.43



Gambar 4. 43 *Uninstall* Apache2

Pada gambar 4.43 terlihat halaman form *uninstall* untuk menguninstal Apache2 dari *node*1. Pada gambar tersebut sudah memilih *server* dan *service* yang akan di *uninstall*.

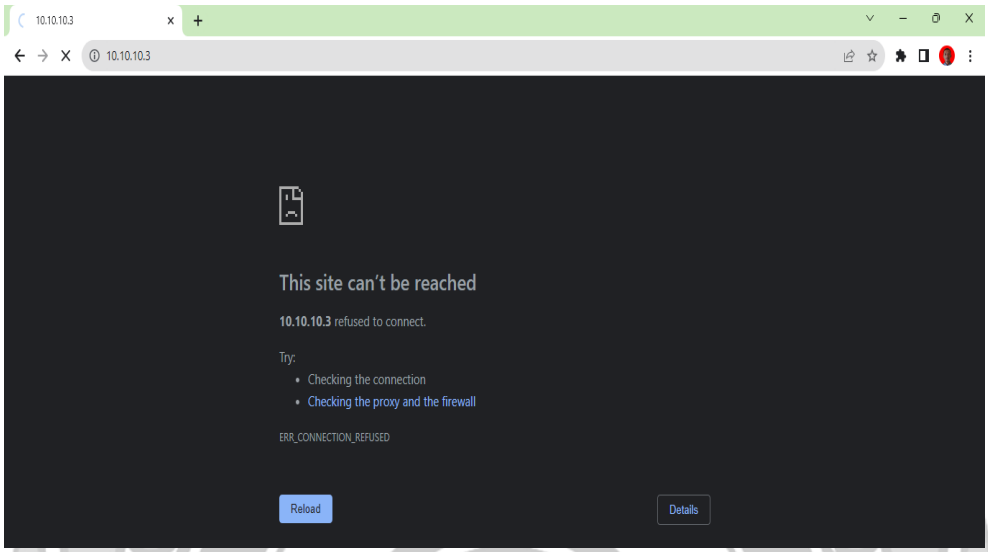
Hasil Setelah uninstal Apache2 pada gambar 4.44



Gambar 4. 44 Hasil Setelah Uninstal Apache2

Pada gambar 4.44 terlihat pada halaman *service* yang dimana *service* Apache2 pada *node*1 telah di *uninstall*.

Testing halaman Web Apache2 dapat dilihat pada gambar 4.45



Gambar 4. 45 Testing Halaman web Apache2 pada browser

Pada gambar 4.45 terlihat halaman web apache2 yang terinstal pada *node*1 sudah di *uninstall*.

4.5 Pengujian Sistem

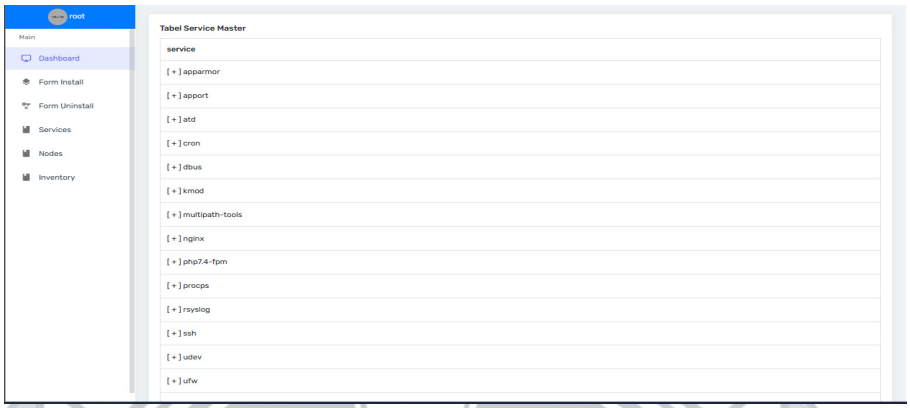
Pengujian ini menggunakan metode *white box* dan *black box* untuk melakukan pengujian terhadap fungsional aplikasi yang telah dibuat

4.5.1 Pengujian *White Box*

Pengujian yang didasarkan pada detail prosedur dan alur logika kode program. Pada kegiatan *whitebox testing*, tester melihat *source code program* dan menemukan *bugs* dari kode program yang diuji. Intinya *white box testing* adalah pengujian yang dilakukan sampai kepada detail pengecekan kode program.

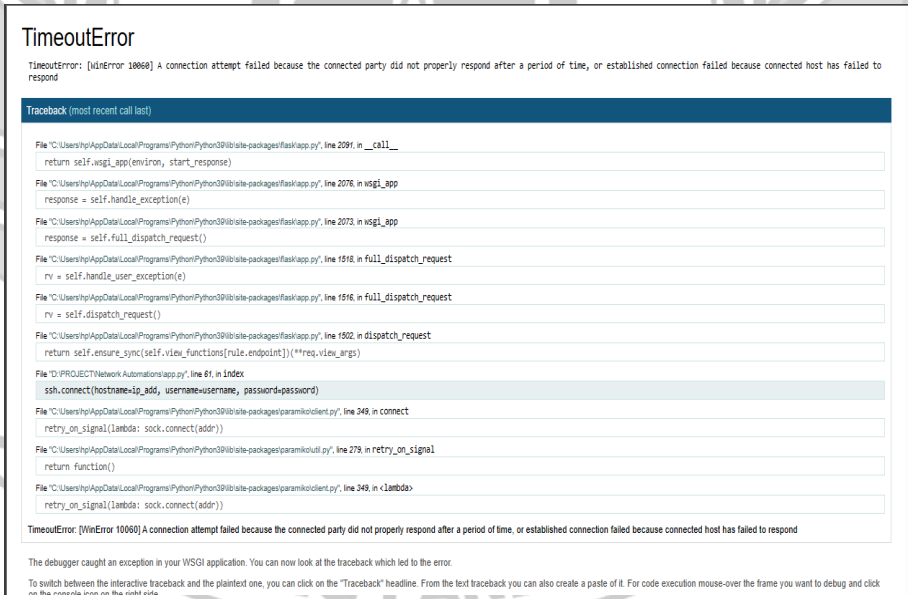
Kegiatan Tester:

a. Test *login* dengan *username*, *ip address* dan *password* yang benar



Gambar 5. 1 Hasil Setelah *Login* dengan *Server* Hidup

Pada gambar 5.1 adalah hasil ketika us*ername, ip address* dan *password* dengan keadaan master *server* yang hidup.



Gambar 5. 2 Hasil Setelah *Login* dengan *Server* Mati

Pada gambar 5.2 terlihat hasil ketika *login* dengan *Ip Address, username* dan *password* yang benar dengan keadaan *server* mati.

```
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    if check_server_status(ip_add, username, password):
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        ssh.connect(hostname=ip_add, username=username, password=password)
        cmd = "service --status-all | grep +"
        # Jalankan perintah pada host target
        stdin, stdout, stderr = ssh.exec_command(cmd)
        output = stdout.read().decode()
        daftar_layanan = output.split('\n')
        ssh.close()
        return redirect(url_for('dashboard'))
    else:
        flash('Server Masih Dalam Keadaan Mati', 'error')
        return render_template('vertical/pages-login.html')
```

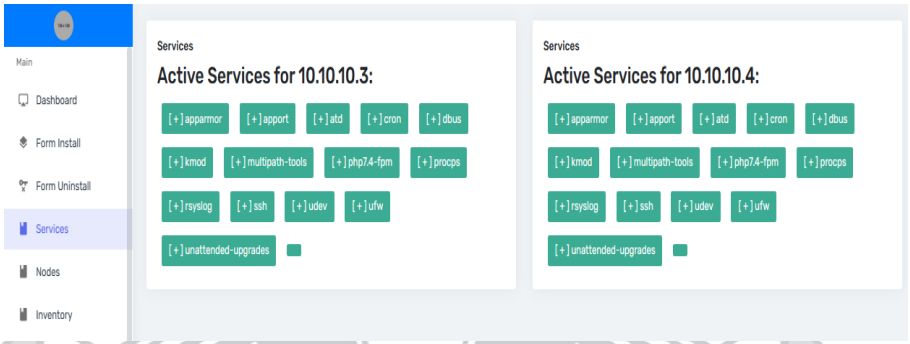
Gambar 5. 3 Penambahan Kode *Handling Error*

Pada gambar 5.3 adalah kode yang ditambahkan untuk mengatasi error pada gambar 5.2. Kode program tersebut akan mengarahkan ke halaman *login* kembali apabila keadaan *server* masih mati. Hasil yang ditampilkan dapat dilihat pada gambar 5.4.

Gambar 5. 4 Gambar Setelah Kode ditambahkan

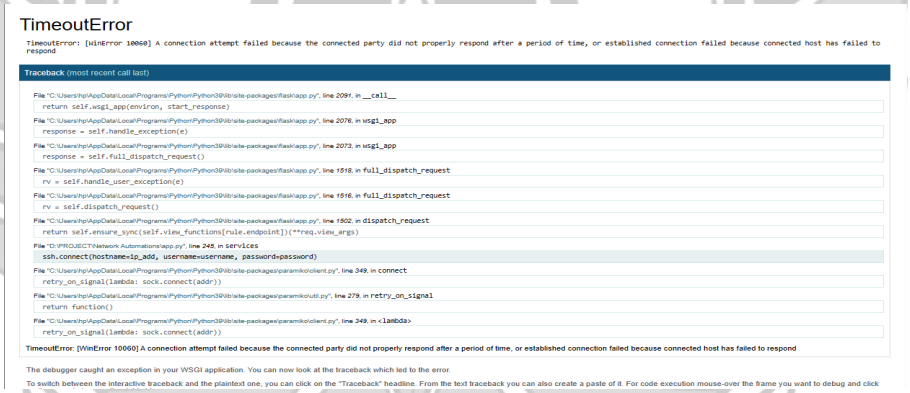
Pada gambar 5.4 terlihat hasil setelah kode ditambahkan dan *error* pada gambar 5.2.

b. Test Halaman *Service*



Gambar 5. 5 *Node* dalam Keadaan Hidup

Pada gambar 5.5 adalah tampilan *service* ketika *node* dalam keadaan hidup.



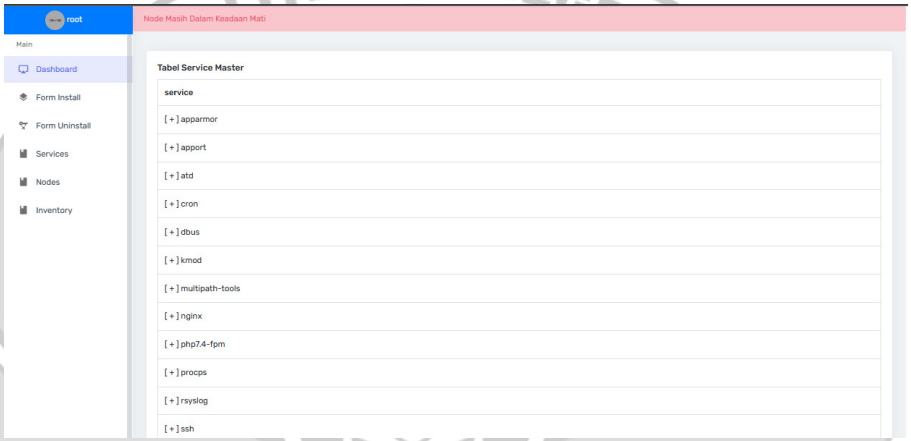
Gambar 5. 6 Hasil Ketika *Node* Mati

Pada gambar 5.6 adalah hasil ketika *node* mati maka halaman *service*-nya error.

```
try:
    if check_server_status(ip_add, username, password):
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        ssh.connect(hostname=ip_add, username=username, password=password)
        cmd = "service --status-all | grep +"
        # Jalankan perintah pada host target
        stdin, stdout, stderr = ssh.exec_command(cmd)
        output = stdout.read().decode()
        daftar_port = output.split('\n')
        ssh.close()
        active_ports_by_server[ip_add] = daftar_port

    else:
        flash('Server Masih Dalam Keadaan Mati', 'error')
        return redirect(url_for('dashboard'))
```

Gambar 5. 7 Tambahan Kode

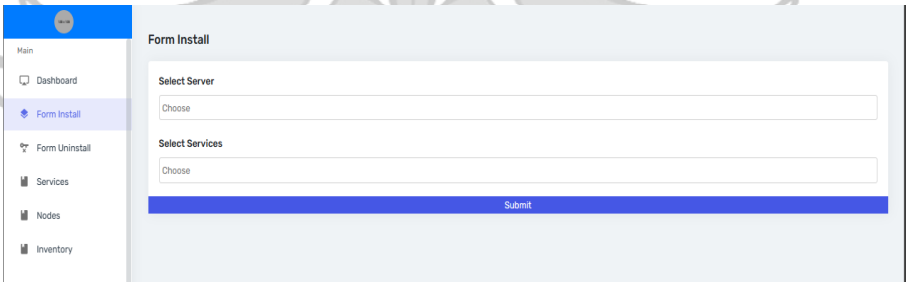
Pada gambar 5.7 adalah kode tambahan untuk meniadakan *error* pada gambar 5.6. Tambahan kode ini berfungsi untuk meredirect kembali ke halaman *Dashboard* dengan menampilkan error ketika keadaan *node server* mati. Hasil yang akan ditampilkan dapat dilihat pada gambar 5.8



Gambar 5. 8 Hasil setelah Kode ditambahkan

Pada Gambar 5.8 terlihat kode tersebut mengarahkan ke *Dashboard* dengan *alert* seperti pada gambar.

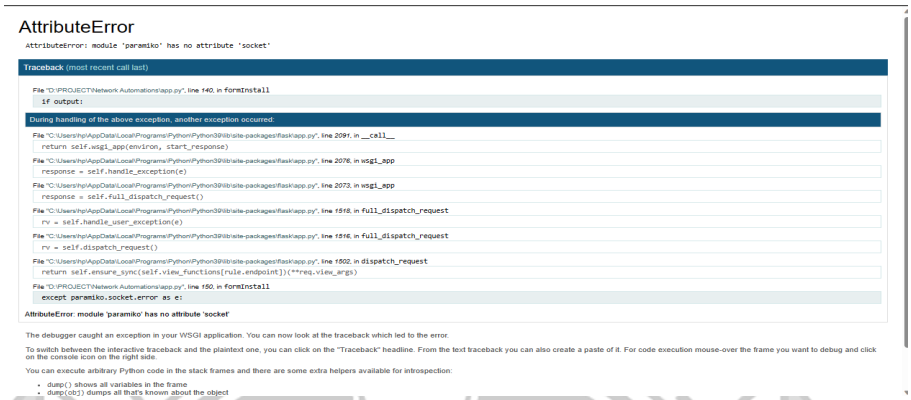
c. Test *Form* Instal dan *Uninstall*



Gambar 5. 9 Form Instal Diisi kosong

Pada gambar 5.9 adalah tampilan *form install* yang diisi kosong. Tidak memilih *server* ataupun *service* maka tampilannya akan seperti pada

gambar 5.10



Gambar 5. 10 Hasil Ketika Form Tidak Diisi

Pada gambar 5.10 terlihat hasil ketika *form install* tidak diisi dan dilakukan submit maka hasilnya *error*.

Penambahan kode untuk mengatasi error pada gambar 5.10, maka ditambahkan kode pada gambar 5.11

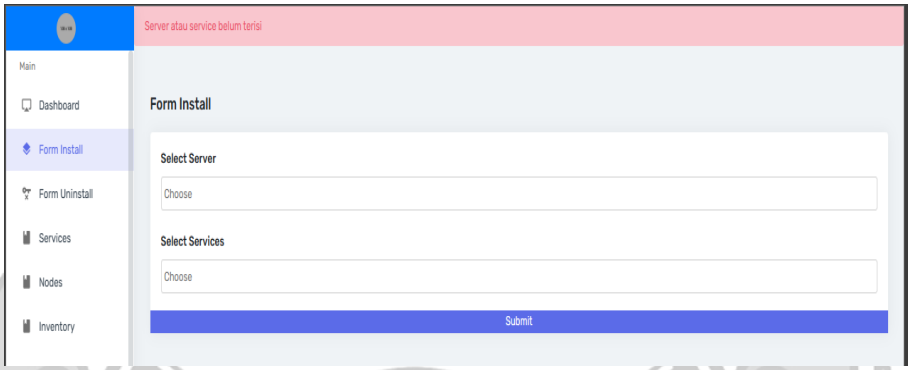
```
if request.method == 'POST':
    selected_servers = request.form.getlist('selected_servers[]')
    selected_services = request.form.getlist('selected_services[]')
    if not selected_servers and selected_services:
        flash("Server atau service belum terisi", "error")
        return redirect(url_for('formUninstall'))
    if not selected_servers:
        flash("Server atau service belum terisi", "error")
        return redirect(url_for('formUninstall'))
    if not  selected_services:
        flash("Server atau service belum terisi", "error")
        return redirect(url_for('formUninstall'))
    try:
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        ssh.connect(hostname=ip_add, username=username, password=password)
        # command = "cd /home/muhtaufikw/"
        # stdin, stdout, stderr = ssh.exec_command(command)
```

Gambar 5. 11 Kode Tambahan

Pada gambar 5.11 terlihat kode yang mengecek ketika *server* dan *service* pada *form install* dan form *uninstall* kosong maka akan diarahkan

kembali ke halaman *form install.*

Ketika kode sudah ditambahkan maka hasil yang akan keluar seperti pada gambar 5.12.



Gambar 5. 12 Hasil Ketika Kode *Handling* Sudah ditambahkan

Pada gambar 5.12 terlihat hasil yang dikeluarkan ketika pada *form install* l dan *uninstall* tidak terisi inputan *server* atau *service*nya.

4.5.2 Pengujian Fungsional

Pengujian fungsional dilakukan dengan metode *black box*. Pengujin ini digunakan untuk memastikan semua proses pada aplikasi berjalan sesuai yang diharapkan. Adapun hasil dari pengujian menggunakan *black box* yaitu sebagai berikut.

Tabel 4. 1 Pengujian *Login*

Data Masukan	Diharapkan	Pengamatan	Kesimpulan	Aspek Pengujian
Ip address, *username* dan *password*	Masuk ke halaman *Dashboard*	Diarahkan ke halaman *Dashboard*	Berhasil	*Login* dan Autentikasi

yang benar				
Ip address, *username* dan *password* salah	Kembali ke halaman *login*	Kembali ke halaman *login* dengan *alert error*	Berhasil	*Login* dan Autentikasi

Tabel 4. 2 Pengujian Fitur Aplikasi

Data Masukan	Diharapkan	Pengamatan	Kesimpulan	Aspek Pengujian
Pengisian pada menu *form install* dan berhasil	Diarahkan ke halaman *service* untuk cek *service* dengan *alert* berhasil	Diarahkan ke halaman *service* dengan *alert* berhasil	Berhasil	Instalasi *Service*
Pengisian form *uninstall* dan berhasil	Diarahkan ke *service* dengan *alert* berhasil	Diarahkan ke *service* dengan *alert* berhasil	Berhasil	Instalasi *Service*
Penambahan *node*s	Diarahkan ke *node*s kembali dengan data yang sudah tertambah	Diarahkan ke *node*s kembali dengan data yang sudah tertambah	Berhasil	Penambahan *Node*
Pengisian form *install* yang kosong	Diarahkan kembali ke halaman *form install* dan tidak dapat menginstal	Diarahkan kembali ke halaman *form install* dan tidak dapat menginstal	Berhasil	Instalasi *Service*

	service	*service*		
Pengisian *form uninstall* yang kosong	Diarahkan kembali ke halaman *form uninstall* dan tidak dapat menginstal *service*	Diarahkan kembali ke halaman *form uninstall* dan tidak dapat menginstal *service*	Berhasil	Instalasi *Service*
Penambahan data pada file *inventory*	Diarahkan ke menu *inventory* dengan data tertambah	Diarahkan ke menu *inventory* dengan data tertambah	Berhasil	*Inventory*

Pengujian koordinasi antara web dengan *server* dilakukan untuk mengecek apakah web dan *server* saling terhubung.

Tabel 4. 3 Pengujian Koordinasi

Data Masukan	Diharapkan	Pengamatan	Kesimpulan	Aspek Pengujian
Ip address, *username* dan *password* yang benar tapi master *server* dalam keadaan mati	Diarahkan kembali ke *login* dengan *alert*	Diarahkan kembali ke *login* dengan *alert*	Berhasil	Kondisi Master
Klik Menu *service* pada saat ada *node*	Kembali ke halaman *Dashboard*	Kembali ke halaman *Dashboard*	Berhasil	Kondisi *Node*

mati	dengan *alert*	dengan *alert*		

4.5.3 Pengujian Perbandingan Instalasi

Pengujian Perbadingan bertujuan untuk melihat seberapa efisiensi aplikasi ini diterapkan secara langsung pada server. Untuk hasil dari pengujian dapat dilihat pada tabel 4.4

Tabel 4. 4 Tabel Pengujian Perbandingan Instalasi

Jumlah Node Server	Instalasi Melalui Aplikasi	Instalasi Manual
1 Buah	13 detik	32 detik
2 Buah	21 detik	1 menit
3 Buah	29 detik	1menit 27 detik
4 Buah	35 detik	1 menit 58 detik
5 Buah	44 detik	2 menit 27 detik
10 Buah	1 menit 17 detik	5 menit

Pengujian yang dilakukan adalah instalasi Nginx Service mulai dari update package, lalu install Nginx dan mengaktifkan Library php yang sudah terinstal pada node server. Dari hasil pengujian diatas ketika menggunakan aplikasi untuk menginstal 1 server lebih cepat daripada kita menginstal manual. Semakin banyak server yang diinstal menggunakan aplikasi maka durasi untuk instal service semakin berkurang.

BAB V PENUTUP

5.1 Kesimpulan

Hasil dari implementasi dan pengujian aplikasi menunjukkan bahwa proses instalasi menggunakan aplikasi jauh lebih efisien daripada metode manual untuk layanan yang ada pada node-server. Kecepatan eksekusi meningkat secara signifikan, menghemat waktu dan tenaga. Selain itu, tingkat akurasi yang lebih tinggi dan konsistensi yang terjaga dari setiap instalasi melalui aplikasi memberikan keunggulan tambahan. Seluruh konfigurasi dan dependensi diatur dengan benar, mengurangi risiko kesalahan manusia. Terakhir, aplikasi juga menyediakan alat pengelolaan yang memudahkan pemantauan dan pemeliharaan layanan pada node-server. Oleh karena itu, direkomendasikan untuk menggunakan aplikasi ini sebagai metode instalasi utama guna memaksimalkan efisiensi dan konsistensi dalam mengelola layanan pada node-server.

5.2 Saran

Adapun saran untuk pengembangan aplikasi *Network Automation* ini yaitu:

1. Penelitian selanjutnya diharapkan dapat mengembangkan program *Ansible* yang dapat *degenerate.*
2. Menambah fitur lainnya agar instalasi bisa lebih detail.
3. Menambahkan fitur *remote SSH* agar bisa langsung tambah *node* tanpa harus konfigurasi *SSH* terlebih dahulu.

DAFTAR PUSTAKA

Affandi, M. R., Hatta, P., & Efendi, A. (2020). Otomatisasi Perangkat Jaringan Komputer Menggunakan *Ansible* Pada Laboratorium Komputer. *SMARTICS Journal*, *6*(2), 48–53.

Ardiansyah, M. (2019). *LKP: Remote Web Server Berbasis Jaringan Cisco Catalyst Series 2960 Menggunakan SSH di PT. Telekomunikasi Indonesia Divre V Jatim*. Universitas Dinamika.

Asfihan, A. (2022). *QoS Adalah*. July 29, 2022. https://adalah.co.id/qos/

Djamaluddin, M. (2021). *Membangun aplikasi untuk memprediksi keberlanjutan studi mahasiswa s1 universitas hasanuddin menggunakan neural network pada micoframework Flask*. Universitas Hasanuddin.

Fahmi, M., Maisyaroh, M., Komarudin, I., Faizah, S., & Fadhilah, I. (2021). Otomatisasi Jaringan Menggunakan *Script Flask* Untuk Penyediaan Konfigurasi Internet Dan Manajemen Mikrotik. *BINA INSANI ICT JOURNAL*, *8*(1), 53–62.

Karki, S., & V, D. K. (2021). Performance Comparison of SSH Libraries. *Journal of University of Shanghai for Science and Technology*, *23*(06), 868–873. https://doi.org/10.51201/jusst/21/05357

Khumaidi, A. (2021). IMPLEMENTATION OF DEVOPS METHOD FOR AUTOMATION OF *SERVER* MANAGEMENT USING *ANSIBLE* . *Jurnal Transformatika*, *18*(2), 199–209.

Mustofa, Z. (2021). *Apa Itu Server? Ketahui Definisi, Fungsi, Manfaat, Jenis-Jenis, Dan Cara Kerjanya*. http://teknik-informatika-

s1.stekom.ac.id/informasi/baca/-Apa-Itu-*Server*-Ketahui-Definisi-Fungsi-Manfaat-Jenis-Jenis-dan-Cara-Kerjanya/e6549306ad041a692cb3143eb8ef7d59092d3479

Suwito, S. (2021). *Penggunaan Bahasa Pemrograman Flask untuk Membuat Aplikasi yang Dapat Digunakan Sebagai Media Dalam Mengajarkan Konsep Perilaku Lentur Balok Beton Bertulang*.

Vassa, M. (2021). *Analisis Performansi Load Balancing Web Server Menggunakan Algoritma Weighted Round Robin Pada Proxmox Ve*. Institut Teknologi Telkom Purwokerto.

Wahid, A. A. (2020). Analisis Metode Waterfall Untuk Pengembangan Sistem Informasi. *J. Ilmu-Ilmu Inform. Dan Manaj. STMIK, No. November*, 1–5.

Yudha, F. (2020). *Monitoring Aktivitas User pada System dengan Menggunakan EFK (Elasticsearch, Fluentd, Kibana) Stack*.