

**PERBANDINGAN WAKTU EKSEKUSI  
OTOMASI MANAJEMEN KONFIGURASI  
SISTEM OPERASI GNU/LINUX ANTARA  
ANSIBLE DENGAN NIXOS**

**PROPOSAL SKRIPSI**



Disusun oleh:  
**M. Rizqi R (20051204034)**  
Dosen Pembimbing:  
**Agus Prihanto, S.T., M.Kom.**

**UNIVERSITAS NEGERI SURABAYA  
FAKULTAS TEKNIK  
PROGRAM STUDI TEKNIK INFORMATIKA  
2024**

## KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa yang telah melimpahkan rahmat dan hidayah-Nya sehingga proposal penelitian dengan judul “Perbandingan Waktu Eksekusi Otomasi Manajemen Konfigurasi Sistem Operasi GNU/Linux Antara Ansible dengan NixOS” ini dapat terselesaikan. Penulis juga mengucapkan terima kasih kepada seluruh yang telah membantu dalam pembuatan proposal penelitian ini.

1. Kedua orang tua atas segala bantuan, bimbingan, dorongan serta doa restu yang diberikan.
2. Bapak Agus Prihanto, S.T., M. Kom. selaku dosen pembimbing yang telah memberikan arahan dan bimbingan dalam penyusunan proposal ini,.
3. Teman-teman Mahasiswa yang telah membantu dalam pengumpulan data dan informasi
4. Universitas Negeri Surabaya yang telah menyediakan fasilitas dan sarana prasarana yang diperlukan dalam penyusunan proposal ini.

Kami menyadari bahwa proposal penelitian ini masih jauh dari sempurna. Oleh karena itu, kami mengharapkan kritik dan saran yang membangun dari para pembaca. Akhir kata, kami berharap proposal penelitian ini dapat bermanfaat bagi para pembaca.

penulis

## ABSTRAK

### PERBANDINGAN WAKTU EKSEKUSI OTOMASI MANAJEMEN KONFIGURASI SISTEM OPERASI GNU/LINUX ANTARA ANSIBLE DENGAN NIXOS

Nama : M. Rizqi R  
NIM : 20051204034  
Program Studi : S1 Teknik Informatika  
Fakultas : Teknik  
Nama Lembaga : Universitas Negeri Surabaya  
Pembimbing : Agus Prihanto, S.T, M.Kom

Pentingnya melakukan manajemen konfigurasi dengan tujuan untuk menghindari penulisan manual konfigurasi sistem operasi secara manual setiap kali menyiapkan sistem operasi. Manajemen konfigurasi juga digunakan untuk mencapai konsistensi dalam setiap kali penerapan sehingga hasil akhir yang diinginkan akan sama setiap kali dilakukan. Terdapat beberapa *tool* untuk melakukan manajemen konfigurasi sistem operasi, terutama untuk sistem operasi berbasis GNU/Linux. Ansible dan NixOS menjadi dua dari banyak pilihan untuk melakukan manajemen sistem operasi. Keduanya memiliki tujuan memudahkan proses manajemen konfigurasi dan memastikan hasil akhir yang diinginkan akan sama setiap kali eksekusi. Sebagai *tool* manajemen konfigurasi, Ansible memiliki waktu eksekusi lebih cepat dari NixOS dengan *nixos-rebuild*. Namun dari segi deklaratif, NixOS terbukti lebih deklaratif karena semua yang ada dalam konfigurasi di manifestasi dalam sistem. Berbeda dengan Ansible yang hanya mengerjakan apa yang ada dalam Ansible Playbook dan tidak menjadi manifestasi dari sistem.

**Kata Kunci** - Ansible, NixOS, *declarative*, *imperative*, Manajemen Konfigurasi

## ABSTRACT

### OPERATING SYSTEM CONFIGURATION MANAJEMEN TIME EXECUTION COMPARISON BETWEEN NIXOS AND ANSIBLE

Author : M. Rizqi R  
NIM : 20051204034  
Study Program : Bachelor's Degree of Informatics Engineering  
Faculty : Engineering  
Institution Name : State University of Surabaya  
Advisor : Agus Prihanto, S.T, M.Kom

The importance of performing configuration management with the purpose of avoiding writing the operating system configuration manual operating system configuration manual every time setting up the operating system. Configuration management is also used to achieve consistency in each application so that the desired so that the desired end result will be the same every time it is done. There are several tools to perform operating system configuration management operating system, especially for GNU/Linux-based operating systems. Ansible and NixOS are two of the many choices for operating system management. to do operating system management. Both have goal of easing the configuration management process and ensure the desired end result will be the same every time execution. As a configuration management tool, Ansible has faster execution time than NixOS with nixos-rebuild. However, from a declarative point of view, NixOS proved to be more declarative because everything in the configuration is manifested in the system. system. Unlike Ansible which only does what is in the Ansible Playbook and does not become a manifestation of the system. manifestation of the system.

**Kata Kunci** - Ansible, NixOS, *declarative*, *imperative*, Configuration Management

## DAFTAR ISI

<b>ABSTRAK</b> .....	<b>ii</b>
<b>ABSTRACT</b> .....	<b>iii</b>
<b>DAFTAR TABEL</b> .....	<b>vi</b>
<b>DAFTAR GAMBAR</b> .....	<b>vii</b>
<b>BAB I PENDAHULUAN</b> .....	<b>1</b>
A    Latar Belakang . . . . .	1
B    Rumusan Masalah . . . . .	2
C    Tujuan . . . . .	2
D    Manfaat . . . . .	2
E    Batasan Masalah . . . . .	3
<b>BAB II KAJIAN PUSTAKA</b> .....	<b>4</b>
A    Penelitian Terdahulu . . . . .	4
B    Manajemen Konfigurasi . . . . .	9
C    Imperatif . . . . .	9
D    Deklaratif . . . . .	9
E    Immutable Distro . . . . .	10
F    Reproducible . . . . .	10
G    Repeatable . . . . .	10
H    NixOS . . . . .	10
I    Ansible . . . . .	10
J    YAML . . . . .	10
K    BASH . . . . .	10
L    NixOS Module . . . . .	11
M    Flake . . . . .	11
N    Home Manager . . . . .	11

O	Virtualisasi . . . . .	11
<b>BAB III</b>	<b>METODE PENELITIAN . . . . .</b>	<b>12</b>
A	Metode Penelitian . . . . .	12
1	Identifikasi Masalah . . . . .	12
2	Studi Literatur . . . . .	13
3	Analisis Kebutuhan . . . . .	13
4	Perancangan Sistem . . . . .	13
5	Implementasi dan Pengujian . . . . .	14
6	Analisis Hasil . . . . .	15
7	Kesimpulan . . . . .	15
B	Analisis Kebutuhan . . . . .	15
1	Kebutuhan perangkat keras ( <i>hardware</i> ) . . . . .	16
2	Kebutuhan perangkat lunak ( <i>software</i> ) . . . . .	16
C	Perancangan Sistem . . . . .	17
1	Perancangan Virtual Machine . . . . .	17
2	Perancangan Topologi . . . . .	18
D	Perancangan Pengujian . . . . .	18
<b>BAB IV</b>	<b>HASIL PENELITIAN DAN PEMBAHASAN . . . . .</b>	<b>21</b>
A	Implementasi . . . . .	21
1	Persiapan Lingkungan NixOS . . . . .	21
2	Struktur Konfigurasi NixOS . . . . .	24
3	Konfigurasi inti configuration.nix . . . . .	27
4	Home Manager . . . . .	28
5	Konfigurasi Ansible . . . . .	29
B	Pengujian . . . . .	31
1	Pasca Install . . . . .	31
2	Penerapan ulang setelah pasca install. . . . .	33
3	Penambahan Paket . . . . .	35
4	Penghapusan Paket . . . . .	37
5	Grafik Perbandingan Waktu . . . . .	39
<b>BAB V</b>	<b>KESIMPULAN DAN SARAN . . . . .</b>	<b>43</b>
A	Kesimpulan . . . . .	43
B	Saran . . . . .	44
	<b>DAFTAR PUSTAKA . . . . .</b>	<b>44</b>

## DAFTAR TABEL

Tabel 2.1 Penelitian Terdahulu .....	6
Tabel 3.1 Kebutuhan perangkat lunak .....	16
Tabel 3.2 Spesifikasi Instance .....	17
Tabel 4.1 nixos-rebuild first target .....	32
Tabel 4.2 nixos-rebuild second target .....	32
Tabel 4.3 nixos-rebuild first target .....	32
Tabel 4.4 nixos-rebuild second target .....	32
Tabel 4.5 ansible first target .....	33
Tabel 4.6 ansible second target .....	33
Tabel 4.7 ansible multi target .....	33
Tabel 4.8 nixos-rebuild first target .....	34
Tabel 4.9 nixos-rebuild second target .....	34
Tabel 4.10 Ansible first target .....	34
Tabel 4.11 Ansible first target .....	35
Tabel 4.12 Ansible first target .....	35
Tabel 4.13 nixos-rebuild first target .....	35
Tabel 4.14 nixos-rebuild second target .....	36
Tabel 4.15 nixos-rebuild first target .....	36
Tabel 4.16 nixos-rebuild second target .....	36
Tabel 4.17 ansible first target .....	36
Tabel 4.18 ansible first target .....	37
Tabel 4.19 ansible multi target .....	37
Tabel 4.20 nixos-rebuild first target .....	37
Tabel 4.21 nixos-rebuild second target .....	38
Tabel 4.22 ansible first target .....	38
Tabel 4.23 ansible second target .....	38
Tabel 4.24 ansible multi target .....	38

## DAFTAR GAMBAR

3.1	Daigram <i>experimental design</i> . . . . .	12
3.2	<i>flowchart</i> manajemen konfigurasi . . . . .	14
3.3	Topologi Pengujian . . . . .	18
3.4	Alur Penerapan Konfigurasi NixOS . . . . .	19
3.5	Alur Penerapan Konfigurasi Ansible . . . . .	20
4.1	NixOS 23.11 installer GNOME . . . . .	22
4.2	NixOS 23.11 Location . . . . .	22
4.3	NixOS 23.11 Keyboard . . . . .	23
4.4	NixOS Partition . . . . .	23
4.5	NixOS 23.11 tty . . . . .	24
4.6	Directory-tree . . . . .	25



## **BAB I**

### **PENDAHULUAN**

#### **A. Latar Belakang**

Pentingnya melakukan manajemen konfigurasi dengan tujuan untuk menghindari penulisan manual konfigurasi sistem operasi secara manual setiap kali menyiapkan sistem operasi. Manajemen konfigurasi juga digunakan untuk mencapai konsistensi dalam setiap kali penerapan sehingga hasil akhir yang diinginkan akan sama setiap kali dilakukan.

Terdapat beberapa *tool* untuk melakukan manajemen konfigurasi sistem operasi, terutama untuk sistem operasi berbasis GNU/Linux. Ansible dan NixOS menjadi dua dari banyak pilihan untuk melakukan manajemen sistem operasi. Keduanya memiliki tujuan memudahkan proses manajemen konfigurasi dan memastikan hasil akhir yang diinginkan akan sama setiap kali eksekusi.

Ansible adalah perangkat lunak otomatisasi TI baris perintah yang ditulis dalam bahasa Python. Aplikasi ini dapat mengonfigurasi sistem, menerapkan perangkat lunak, dan mengatur alur kerja tingkat lanjut untuk mendukung penerapan aplikasi, pembaruan sistem, dan banyak lagi (Ansible, 2016).

Ansible memungkinkan kita mendeklarasikan konfigurasi sistem kita dalam sebuah Ansible Playbook. Ansible Playbook akan dijalankan pada sebuah sistem yang telah memiliki sistem operasi. Konfigurasi Ansible Playbook ditulis menggunakan format yml yang merupakan format khusus untuk konfigurasi baik sistem maupun aplikasi. Ansible akan menjalankan setiap perintah pada sistem operasi yang telah di install secara otomatis satu per satu. Ansible memungkinkan kita melakukan setup banyak sistem sekaligus dengan konfigurasi yang telah ada. Diharapkan dari Ansible adalah sistem-sistem yang terdaftar memiliki hasil akhir yang sama.

NixOS adalah sistem operasi berbasis GNU/Linux

yang dibangun dengan Nix build system (NixOS, 2023) . NixOS menggunakan file dalam format “.nix” yang disebut sebagai NixOS module untuk mendeklarasikan sebuah sistem. Dalam file tersebut terdapat seluruh konfigurasi sistem mulai dari *bootloader*, *packages*, *users*, *system services*.

Apa yang tertulis dalam module tersebut adalah manifestasi dari sistem yang dideklarasikan menggunakan bahasa nix yang merupakan bahasa pemrograman fungsional. Ini menghasilkan konfigurasi sistem operasi yang *reproducible* sehingga dapat digunakan berkali-kali pada waktu yang berbeda dan menghasilkan manifestasi yang tetap.

Banyak kelebihan dan beberapa kekurangan yang dimiliki oleh metode deklaratif dari NixOS dan metode campuran (imperatif dan deklaratif) dari Ansible. Berdasarkan landasan tersebut, maka penulis ingin meneliti dan membandingkan dari segi performa waktu eksekusi yang dibutuhkan oleh masing-masing *tool* manajemen konfigurasi.

## **B. Rumusan Masalah**

Adapun rumusan masalah berdasarkan latar belakang diatas yaitu:

1. Bagaimana membuat sistem GNU/Linux yang terdeklarasikan menggunakan *tool* manajemen konfigurasi.
2. Berapa lama waktu eksekusi *tool* manajemen konfigurasi yang diimplementasikan.

## **C. Tujuan**

Adapun tujuan dalam penelitian ini adalah:

1. Membuat sistem GNU/Linux yang terdeklarasikan menggunakan NixOS dan Ansible
2. Mengukur waktu eksekusi *tool* manajemen konfigurasi NixOS dan Ansible

## **D. Manfaat**

Manfaat yang dapat dihasilkan dari penelitian ini antara lain:

1. Efisiensi waktu dalam melakukan manajemen konfigurasi sistem operasi berbasis GNU/Linux.
2. Konfigurasi sebuah sistem menjadi deklaratif dalam baris kode.
3. Mengurangi terjadinya human error dalam mengerjakan konfigurasi dan tugas yang berulang-ulang.
4. Mendapatkan hasil akhir yang konsisten dari penerapan *tool* manajemen konfigurasi.

#### **E. Batasan Masalah**

Adapun batasan masalah yang digunakan untuk menghindari penyimpangan dari judul dan tujuan adalah sebagai berikut:

1. Tools yang digunakan untuk manajemen konfigurasi adalah Ansible dan NixOS.
2. Kasus studi dalam menggunakan NixOS menggunakan file konfigurasi dasar, flake, dan home-manager.
3. Manajemen konfigurasi yang dilakukan meliputi setup dari sistem kosong ke konfigurasi yang diinginkan oleh penulis.
4. Perbandingan akan dilihat berdasarkan waktu yang dibutuhkan untuk eksekusi penerapan konfigurasi.

## BAB II KAJIAN PUSTAKA

### A. Penelitian Terdahulu

Sudah ada penelitian terdahulu terkait otomasi menggunakan Ansible. Salah satu diantaranya adalah yang dilakukan oleh Thufail Qolba AUFAR yang berjudul *“Configuration Management dengan Ansible dan Telegram Untuk Automasi Laboratorium Komputer di ITIK”* pada tahun 2023 (Thufail Qolba, 2023). Dalam penelitian tersebut menggunakan Ansible sebagai *tool* manajemen konfigurasi laboratorium komputer. Sistem operasi yang digunakan oleh target host adalah Windows dan Ansible Module yang digunakan adalah `win_chocolatey` yang merupakan modul untuk `chocolatey package manager` di Windows.

Sistem Configuration Management dengan Ansible dan Telegram berhasil dibuat sesuai dengan fungsionalitas dan rancangan yang telah dibuat. Hal ini dibuktikan dari pengujian fungsionalitas dengan black box testing yang telah dilakukan bahwa setiap tugas dieksekusi dapat berjalan dengan persentase keberhasilan sebesar 100%.

Penelitian untuk otomasi dan manajemen konfigurasi distribusi GNU/Linux ditulis oleh Putu Hariyadi dan Khairan Marzuki dengan judul *“Implementation Of Configuration Management Virtual Private Server Using Ansible”* pada tahun 2020 (Hariyadi & Marzuki, 2020). Pada penelitian tersebut, Ansible digunakan untuk melakukan otomasi pembuatan container pada PVE (Proxmox Virtual Environment) yang bertujuan untuk menyiapkan lingkungan *high availability* sebagai media praktikum kelompok untuk dosen, mahasiswa, dan asisten laboratorium.

Hasilnya adalah manajemen otomasi VPS secara keseluruhan bekerja dengan baik dan dapat diterapkan di Proxmox Virtual Environment (PVE) cluster. Playbook dapat memulai dan menghentikan containers per kelompok siswa secara dinamis berdasarkan jadwal praktikum.

Penelitian terkait penggunaan NixOS sebagai manajemen konfigurasi telah dijabarkan oleh Kalle Kumpulainen dalam tesis ber judul "*NixOS: Järjestelmäkonfiguraation Hallintaan Erikoistunut Linux-jakelu*" pada tahun 2019. Dalam tesis tersebut dituliskan bagaimana NixOS menjadi sebuah distribusi GNU/Linux yang hampir keseluruhan sistemnya terdeklarasi dalam bentuk konfigurasi. Terkait apa saja perbedaan dibandingkan dengan distribusi GNU/Linux yang telah ada (Kumpulainen, 2019).

Ansible juga pernah diteliti dan dibandingkan dengan metode manajemen konfigurasi konvensional yaitu shell scripting dengan BASH. Penelitian ini dilakukan oleh Tedi Alfandi, T.M Diansyah, dan Risiko Liza yang tertuang dalam "*Analisis Perbandingan Manajemen Konfigurasi Menggunakan Ansible dan Shell Script Pada Cloud Server Deployment AWS*" pada tahun 2020. Didapat kesimpulan bahwa penggunaan tool manajemen konfigurasi dapat memperlengkap pekerjaan dalam membangun *web server* (Alfandi et al., 2020).

Dalam penelitian lain yang dilakukan oleh Muh. Akromi Arya Pratama dan I Putu Hariyadi, dalam sebuah jurnal berjudul "*Otomasi Manajemen dan Pengawasan Linux Container (LXC) Pada Proxmox VE Menggunakan Ansible*". Ansible digunakan untuk mempermudah proses otomasi pembuatan LXC pada proxmox untuk praktikum SMKN 6 Mataram. Kesimpulan yang didapat ialah Ansible mampu melakukan otomasi untuk membuat, menjalankan, menghentikan dan menghapus LXC serta mengatur *user permission* dalam lingkup *batch* (Pratama & Hariyadi, 2021).

*Tabel 2.1: Penelitian Terdahulu*

No.	Judul	Nama Peneliti	Kesimpulan
1	<i>Configuration Management</i> dengan Ansible dan Telegram Untuk Automasi Laboratorium Komputer di JTIK	Thufail Qolba Aufar	Sistem Configuration Management dengan Ansible dan Telegram berhasil dibuat sesuai dengan fungsionalitas dan rancangan yang telah dibuat. Hal Ini dibuktikan dari pengujian fungsionalitas dengan black box testing yang telah dilakukan bahwa setiap tugas dieksekusi dapat berjalan dengan persentase keberhasilan sebesar 100.

No.	Judul	Nama Peneliti	Kesimpulan
2	<i>Implementation Of Configuration Management Virtual Private Server Using Ansible</i>	Putu Hariyadi, Khairan Marzuki	Manajemen otomasi VPS secara keseluruhan bekerja dengan baik dan dapat diterapkan di Proxmox Virtual Environment (PVE) cluster. Playbook dapat memulai dan menghentikan containers per kelompok siswa secara dinamis berdasarkan jadwal praktikum.

No.	Judul	Nama Peneliti	Kesimpulan
3	<i>NixOS: Järjestelmä konfiguraation Hallintaan Erikoistunut Linux jakelu</i>	Kalle Kumpulainen	Distribusi ini pada awalnya dibuat untuk memecahkan masalah asli distribusi perangkat lunak dan manajemen konfigurasi sistem operasi, seperti keamanan dan determinisme fungsi yang diinginkan. Untuk mengatasi masalah ini, NixOS diimplementasikan sejak awal dengan cara yang sangat tidak biasa dibandingkan dengan distribusi Linux terkenal lainnya.
4	Analisis Perbandingan Manajemen Konfigurasi Menggunakan Ansible dan Shell Script Pada Cloud Server Deployment AWS	Tedi Alfiandi, T.M Diansyah, Risko Liza	Penggunaan tool manajemen konfigurasi dapat memperingkas pekerjaan dalam membangun web server



No.	Judul	Nama Peneliti	Kesimpulan
5	Otomasi Manajemen dan Pengawasan Linux Container (LXC) Pada Proxmox VE Menggunakan Ansible	Muh. Akromi Arya Pratama dan I Putu Hariyadi	Ansible mampu melakukan otomasi untuk membuat, menjalankan, menghentikan dan menghapus LXC serta mengatur <i>user permission</i> dalam lingkup <i>batch</i>

#### B. Manajemen Konfigurasi

Manajemen konfigurasi adalah metode dimana sebuah sistem di manajemen menggunakan file-file yang mendeskripsikan apa yang harus dilakukan sistem tersebut. Harapan dari hasil file-file konfigurasi ini adalah konsistensi pada hasil akhir setelah konfigurasi tersebut dijalankan. Metode ini juga bertujuan untuk meningkatkan efisiensi dalam replikasi dan manajemen sistem sehingga administrator tidak melakukan konfigurasi dari nol hingga selesai secara berulang.

#### C. Imperatif

Dalam konteks manajemen konfigurasi, imperatif merupakan metode dimana administrator menetapkan langkah-langkah yang perlu dilakukan sebuah sistem untuk mencapai keadaan tertentu. Dalam kasus penelitian ini adalah Ansible dimana setiap definisi yang kita tulis dalam Ansible Playbook akan dijalankan satu persatu dari awal hingga akhir dengan harapan bahwa apa yang kita definisikan dalam Playbook tercapai.

#### D. Deklaratif

Dalam konteks manajemen konfigurasi, deklaratif merupakan metode dimana administrator menetapkan rincian tentang keadaan akhir sistem dalam file-file konfigurasi. Dalam kasus penelitian ini adalah NixOS

Module yang berisikan rincian hasil akhir yang kita inginkan dalam NixOS. NixOS Module akan di evaluasi oleh Nix build system untuk menggapai tujuan ini.

**E. Immutable Distro**

Immutable Distro adalah kategori distribusi GNU/Linux yang dimana sistem operasi read-only yang tidak mengijinkan modifikasi di root file system. Ini berarti kita tidak bisa dengan mudah memodifikasi OS. Ini termasuk file sistem, berkas, aplikasi, bahkan konfigurasi. Bahkan sebagai administrator, kita tidak bisa memodifikasi distribusi tersebut.

**F. Reproducible**

Dalam konteks manajemen konfigurasi, reproducible merujuk pada hasil akhir yang konsisten setiap kali konfigurasi diterapkan.

**G. Repeatable**

Dalam konteks manajemen konfigurasi, repeatable merujuk pada tahapan-tahapan yang dapat diulang dengan tujuan hasil serupa.

**H. NixOS**

Distribusi sistem operasi GNU/Linux yang terintegrasi dengan Nix package manager dimana konfigurasi sistem operasi dideklarasikan dalam Nix Module.

**I. Ansible**

Ansible adalah perangkat lunak otomatisasi TI baris perintah yang ditulis dalam bahasa Python. Aplikasi ini dapat mengonfigurasi sistem, menerapkan perangkat lunak, dan mengatur alur kerja tingkat lanjut untuk mendukung penerapan aplikasi, pembaruan sistem, dan banyak lagi (RedHat, 2022a)

**J. YAML**

Bahasa serialisasi data yang bisa dibaca oleh manusia. YAML umumnya digunakan untuk file konfigurasi dan aplikasi dimana data disimpan atau ditransmisikan.(Wikipedia)

**K. BASH**

Bourne-Again Shell (BASH) adalah Unix-shell dan bahasa perintah yang biasanya berjalan di jendela text dimana

pengguna mengetik perintah yang mengakibatkan aksi

**L. NixOS Module**

Module yang berisi Nix expression dengan struktur yang spesifik dan digunakan untuk membangun konfigurasi Nix yang utuh.

**M. Flake**

Nix flakes menyediakan cara standar untuk menulis ekspresi Nix (dan juga paket-paket) yang ketergantungannya disematkan dalam file kunci, sehingga meningkatkan kemampuan reproduksi instalasi Nix.

**N. Home Manager**

Home Manager adalah sistem untuk mengelola lingkungan pengguna dengan menggunakan manajer paket Nix. Dengan kata lain, Home Manager memungkinkan Anda menginstall perangkat lunak secara deklaratif pada profil user Anda, daripada menggunakan nix-env, mengelola dotfile di direktori home pengguna Anda.

**O. Virtualisasi**

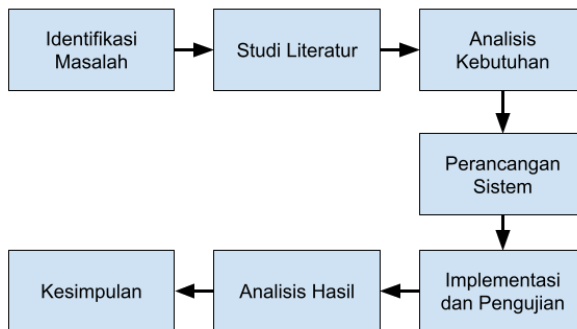
Virtualisasi merupakan proses berbasis perangkat lunak yang membagi komputer tunggal menjadi beberapa mesin virtual / virtual machine, tiap-tiap mesin virtual memiliki sistem operasi dan aplikasi miliknya sendiri (IBM, 2024)

### BAB III METODE PENELITIAN

Metodologi yang digunakan untuk penelitian ini memiliki beberapa tahapan sebagai pedoman agar hasil yang dicapai sesuai dan tidak menyimpang dari tujuan yang telah ditentukan sebelumnya.

#### A. Metode Penelitian

Metode yang diterapkan pada penelitian ini adalah metode experimental design. Penerapan metode bertujuan untuk menganalisa perbandingan manajemen konfigurasi sistem operasi GNU/Linux antara Ansible dengan NixOS.



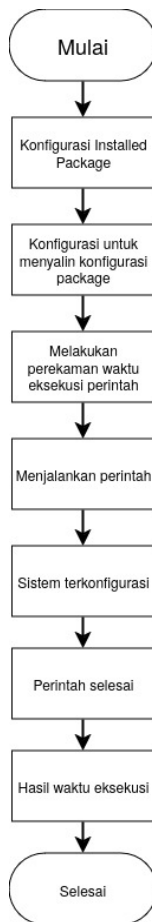
*Gambar 3.1: Daigram experimental design*

Berikut merupakan alur tahapan yang dilakukan dalam penelitian ini. Pada gambar 3.1, dapat diketahui tahapan penelitian yang akan diterapkan pada penelitian ini, yaitu:

1. **Identifikasi Masalah**  
Tahapan awal untuk melakukan penelitian merupakan identifikasi masalah. Pada tahapan ini, permasalahan yang diidentifikasi mengenai manajemen konfigurasi di GNU/Linux untuk mempersingkat waktu konfigurasi dan membuat sistem yang terdeklarasi. Peneliti akan membuat konfigurasi sistem operasi GNU/Linux menggunakan Ansible dan NixOS

dengan kebutuhan yang sama dengan tujuan untuk membandingkan efisiensi kecepatan penerapan *tool* manajemen konfigurasi.

2. Studi Literatur  
Peneliti mencari literatur yang relevan untuk menjadi referensi pendukung dalam penelitian. Literatur yang digunakan dalam penelitian ini berhubungan dengan penerapan manajemen konfigurasi yang didapat dari berbagai macam sumber seperti buku, artikel, jurnal nasional maupun internasional.
3. Analisis Kebutuhan  
Tahap ini bertujuan untuk mendapatkan pemahaman yang menyeluruh dan terperinci tentang kebutuhan utama sistem seperti yang didefinisikan agar tujuan tercapai, yang kemudian secara jelas didefinisikan, ditinjau dan disepakati bersama.
4. Perancangan Sistem  
Perancangan dilakukan dengan menerapkan manajemen konfigurasi pada mesin virtual agar mendapatkan hasil waktu yang dibutuhkan untuk menerapkan konfigurasi. Berikut flowchart proses dari penerapan konfigurasi yang akan dibuat:



*Gambar 3.2: flowchart manajemen konfigurasi*

5. Implementasi dan Pengujian  
Pada tahap ini, rancangan konfigurasi yang sudah dibuat akan diimplementasikan sesuai alur kerja konfigurasi yang telah dibuat pada lingkungan mesin virtual agar lebih efisien dan lebih terisolasi. Instalasi dan konfigurasi dilakukan sesuai referensi dokumentasi dari masing-masing *tool* manajemen konfigurasi. Kustomisasi dari perangkat lunak yang

diperlukan tergantung pada kebutuhan penelitian. Parameter pengujian yang akan dilakukan adalah menguji kecepatan waktu eksekusi dan *reproducibility* dari masing-masing *tool* manajemen konfigurasi.

6. Analisis Hasil

Pada tahap ini, hasil dari implementasi dan pengujian akan dianalisis untuk mengetahui kecepatan penerapan konfigurasi dan konsistensi *tool* manajemen konfigurasi. Untuk mengetahui waktu yang dibutuhkan oleh *tool* manajemen konfigurasi untuk menerapkan konfigurasi, akan digunakan perintah “time”. Untuk mengetahui konsistensi penerapan, akan dilihat berdasarkan versi paket yang di install dalam rentan waktu yang berbeda.

7. Kesimpulan

Tahapan terakhir dalam penelitian ini akan dilakukan penarikan kesimpulan dan pemberian saran dari seluruh penelitian yang telah dilakukan berdasarkan rumusan masalah yang telah dijelaskan sebelumnya. Hasil dari pengujian *tool* manajemen konfigurasi dapat menjadi jawaban dari masalah yang telah dijelaskan, serta data dari penelitian diharapkan berguna dalam efisiensi dan konsistensi penerapan manajemen konfigurasi sistem operasi GNU/Linux dan dapat menjadi referensi penelitian berikutnya.

## B. Analisis Kebutuhan

Analisis kebutuhan merupakan analisis yang dibutuhkan untuk menentukan detail kebutuhan pada penelitian analisis perbandingan manajemen konfigurasi sistem operasi GNU/Linux antara Ansible dengan NixOS. Penelitian ini mengotomasi konfigurasi sistem operasi GNU/Linux dan membuat sistem menjadi terdeklarasi. Sehingga dibutuhkan perangkat-perangkat yang dapat mendukung agar penelitian ini berjalan sesuai tujuan. Berikut merupakan kebutuhan yang dibagi menjadi beberapa bagian, yaitu:

1. Kebutuhan perangkat keras (*hardware*)  
 Perangkat keras yang diperlukan guna tujuan penelitian yaitu Laptop sebagai uji coba dengan spesifikasi berikut:  
 Processor : Intel Core i5 11400H  
 RAM : 24 GB  
 SSD : 1 TB  
 Sistem Operasi : NixOS 24.05 (Uakari) x86\_64
2. Kebutuhan perangkat lunak (*software*)  
 Perangkat lunak berfungsi untuk pengoperasian sistem pada penelitian ini. Pada penelitian ini, sistem operasi yang digunakan berbasis GNU/Linux.

*Tabel 3.1: Kebutuhan perangkat lunak*

No.	Nama Perangkat Lunak	Keterangan
1	Ansible	Alat yang digunakan untuk manajemen konfigurasi
2	NixOS	Sistem Operasi dengan fitur manajemen konfigurasi
3	Ubuntu Server 24.04 LTS	Sistem Operasi yang digunakan untuk target Ansible
4	Nix Flakes	Alat yang digunakan untuk manajemen <i>inputs</i> pada NixOS
5	Home Manager	Alat yang digunakan untuk manajemen konfigurasi level pengguna pada NixOS
6	<i>time</i>	Alat yang digunakan untuk mencatat lama waktu eksekusi perintah



### C. Perancangan Sistem

Perancangan sistem dilakukan untuk membuat desain perencanaan arsitektur sistem yang akan dibangun dapat berjalan sesuai dengan tujuan penelitian.

#### 1. Perancangan Virtual Machine

Perancangan sistem pada penelitian ini secara keseluruhan menggunakan enam *virtual machine*, yang terdiri dari tiga *instance* Ubuntu Server 24.04 LTS dan tiga *instance* NixOS minimal. Satu dari tiga *virtual machine* bersifat sebagai *master* dan dua sebagai *target*.

Tabel 3.2: Spesifikasi Instance

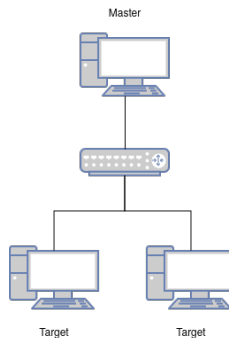
Instance Ansible	
Sistem Operasi	Ubuntu Server 24.04 LTS
Processor	2
Memory	4 GB
Storage	50 GB
Jumlah <i>instance</i>	3
Instance NixOS	
Sistem Operasi	NixOS 23.11
Processor	2
Memory	4 GB
Storage	50 GB
Jumlah <i>instance</i>	3

Alasan perbedaan target dari Ansible dan NixOS adalah karena nixos-rebuild tidak bisa dijalankan pada sistem operasi selain NixOS. Ansible juga tidak dapat digunakan di NixOS karena sifat NixOS sendiri yang *immutable* membuat metode imperatif untuk konfigurasi sistem operasi tidak memungkinkan. Namun untuk mengurangi aspek perbedaan pada

saat *benchmark*, maka akan ditambahkan percobaan dimana target Ansible juga menggunakan *nixpkgs* untuk instalasi paket. Ini dikarenakan *nixpkgs* dapat di *install* di semua distribusi GNU/Linux.

## 2. Perancangan Topologi

Perancangan topologi untuk penelitian ini adalah dengan sistem master dan target. Sistem ini akan membuat satu perangkat sebagai pusat kendali dari beberapa target yang menjadi endpoint dari manajemen konfigurasi.



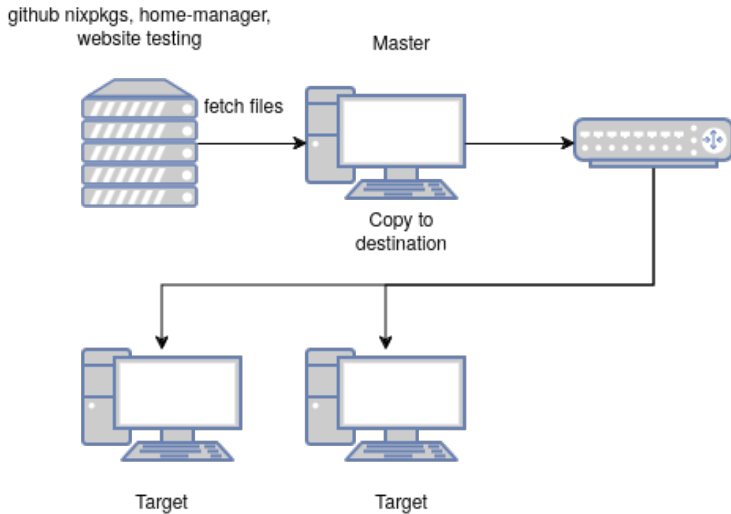
Gambar 3.3: Topologi Pengujian

Dengan topologi ini, diharapkan penerapan manajemen konfigurasi terpusat dan dapat diterapkan di dua mesin target dan memiliki hasil akhir yang sama dan konsisten antar mesin.

## D. Perancangan Pengujian

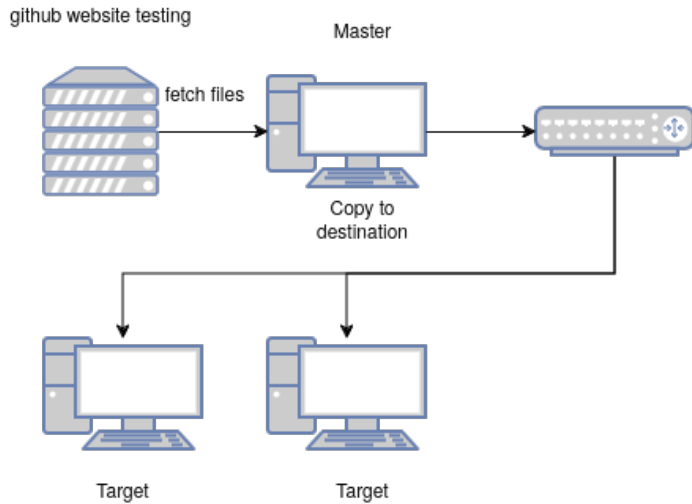
Perancangan pengujian pada penelitian ini akan melakukan instalasi berbagai macam perangkat lunak. Perangkat lunak yang diinstall adalah *tmux*, *vim*, *neovim*, *htop*. Kemudian akan dilakukan beberapa konfigurasi perangkat lunak yaitu konfigurasi untuk *tmux*, *vim*, *neovim*, *openssh*, *firewall*.

Kemudian akan dihadirkan service tambahan yaitu berupa *web server* menggunakan Nginx. Nginx juga akan dikonfigurasi untuk menampilkan *website* sederhana yang kode sumbernya diambil dari *repository* Github.



Gambar 3.4: Alur Penerapan Konfigurasi NixOS

Pada gambar 3.4 merupakan alur penerapan konfigurasi menggunakan NixOS. Master akan melakukan *clone* dari *repository* nixpkgs, home-manager dan website-testing untuk mengambil konfigurasi sistem serta file *website* yang akan di *deploy*. Kemudian *master* akan melakukan build untuk konfigurasi keseluruhan sistem sehingga menghasilkan nix derivation yang membentuk NixOS. Hasil dari derivation tersebut kemudian akan di *copy* dan diterapkan ke dua mesin target.



*Gambar 3.5: Alur Penerapan Konfigurasi Ansible*

Pada gambar 3.5 merupakan alur penerapan konfigurasi menggunakan Ansible. *Master* akan melakukan *clone* dari *repository* github website-testing. Kemudian *master* akan mulai menerapkan konfigurasi untuk target dan menginstall semua paket yang di definisikan dan melakukan konfigurasi yang diinginkan. Setelah konfigurasi sistem selesai diterapkan kemudian Ansible akan menyalin file *website* yang ada di master ke target.

Setiap kali perintah untuk melakukan konfigurasi di eksekusi, waktu eksekusi akan dicatat menggunakan perintah "time". Eksekusi akan dilakukan secara berulang untuk mendapatkan rata-rata waktu yang dibutuhkan untuk menerapkan konfigurasi.

## BAB IV

### HASIL PENELITIAN DAN PEMBAHASAN

Hasil dari penelitian peneliti berupa data-data terkait waktu eksekusi yang dibutuhkan untuk penerapan manajemen konfigurasi dari eksekusi nixos-rebuild dan ansible-playbook. Peneliti membangun dua jenis konfigurasi, yaitu konfigurasi untuk NixOS dan Ansible. Untuk konfigurasi NixOS menggunakan flake untuk mencapai *reproducibility* dan Home-Manager untuk mengkonfigurasi aplikasi pengguna.

Untuk konfigurasi Ansible dibuat semirip mungkin dengan konfigurasi NixOS. Perbedaan nya adalah metode file konfigurasi pada Ansible menggunakan `ansible.posix.synchronize` dan `ansible.builtin.copy` untuk menyalin / mensinkronisasi file konfigurasi setiap aplikasi dan *services*.

#### A. Implementasi

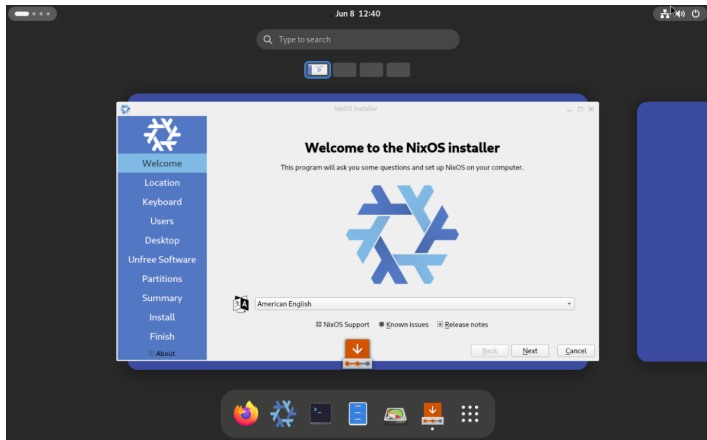
##### 1. Persiapan Lingkungan NixOS

Untuk persiapan lingkungan NixOS baik *master* maupun target adalah sebagai berikut:

##### 1.a. Download ISO dan konfigurasi VM

*Download* ISO NixOS GNOME di <https://nixos.org>, kemudian install dalam VM Proxmox dengan BIOS (OVMF) UEFI q35 dan CPU Type : host agar kita dapat memanfaatkan semua instruction set dari CPU. Lanjutkan dengan spesifikasi CPU, RAM, dan HDD seperti yang telah dicantumkan pada BAB III. Untuk NixOS pada saat penulisan yaitu versi ISO 23.11 belum mendukung *secure boot* secara bawaan. Maka dari itu, perlu untuk menonaktifkan *secure boot* pada VM Proxmox agar ISO bisa dipakai. Saat booting, *spam* tombol 'Esc' untuk masuk ke boot menu dan matikan *Secure Boot* pada bagian *Device Manager* -> *Secure Boot Configuration* -> *Disable secure boot next* -> *Attempt Secure Boot* kemudian klik spasi dan tekan 'y'. Tekan 'F10'

untuk menyimpan konfigurasi. Tekan 'Esc' dua kali untuk keluar dan pilih 'Reset'. Setelah itu, kita akan disambut dengan GNOME *Desktop Environment* seperti berikut:

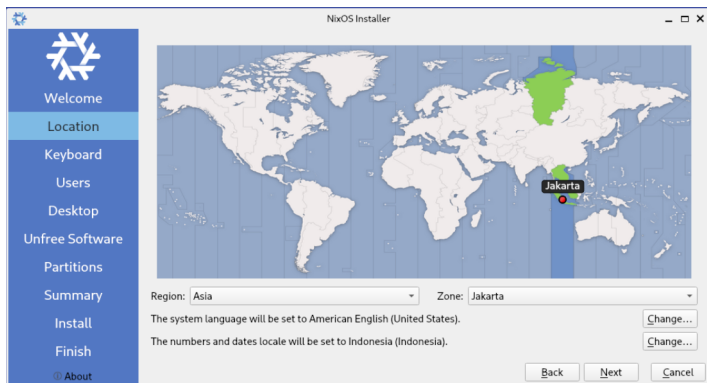


*Gambar 4.1: NixOS 23.11 installer GNOME*

## 1.b. Instalasi

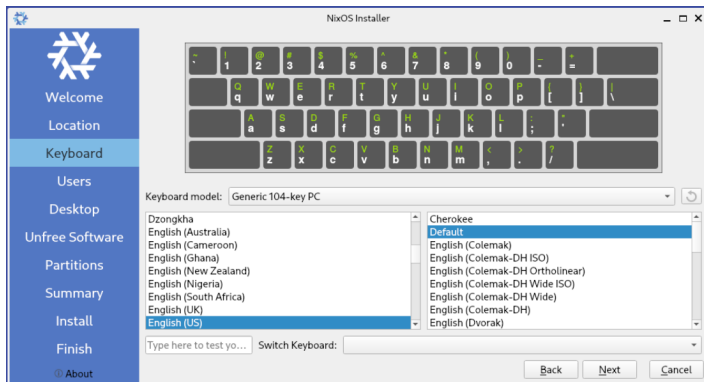
Kemudian kita masuk ke tahap instalasi NixOS sebagai berikut:

### 1. Atur lokasi ke Asia/Jakarta.



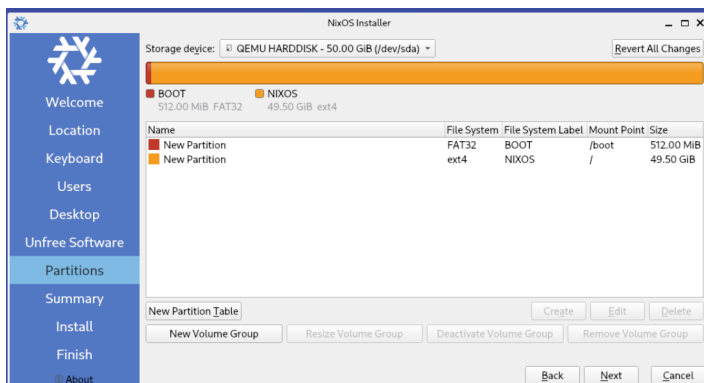
*Gambar 4.2: NixOS 23.11 Location*

## 2. Atur keyboard ke US default.



Gambar 4.3: NixOS 23.11 Keyboard

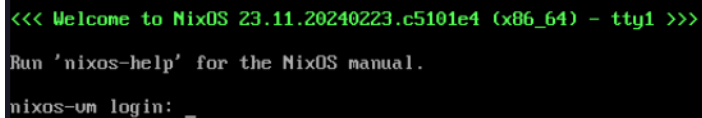
3. Buat username dan password kemudian centang bagian "Use the same password for the administrator account".
4. Untuk pilihan Desktop, pilih "No desktop" karena kita hanya butuh akses SSH.
5. Centang bagian "Allow Unfree Software".
6. Buat skema partisi seperti berikut dengan tabel partisi GPT:



Gambar 4.4: NixOS Partition

## 7. Lanjutkan hingga tahapan install.

8. Ketika proses instalasi, NixOS akan *stuck* tepat pada 46%. Ini karena NixOS sedang melakukan *build derivation* untuk semua paket yang diperlukan.
9. Ketika instalasi sudah selesai, silahkan di *restart*.
10. Setelah instalasi, kita akan disambut dengan *console login* seperti berikut:

A screenshot of a terminal window showing the NixOS 23.11 tty login screen. The text is as follows: <<< Welcome to NixOS 23.11.20240223.c5101e4 (x86\_64) - tty1 >>>, Run 'nixos-help' for the NixOS manual., nixos-vm login: \_</div>
<div data-bbox="383 364 679 385" data-label="Caption">
 Gambar 4.5: NixOS 23.11 tty
</div>
<div data-bbox="244 401 410 423" data-label="Section-Header">
 1.c. Setup SSH
</div>
<div data-bbox="290 432 869 494" data-label="Text">
 Setelah itu, kita aktifkan SSH dan mematikan *firewall* dengan menambahkan baris dibawah ini kedalam file /etc/nixos/configuration.nix:
</div>
<div data-bbox="290 499 836 577" data-label="Text">

```
services.openssh.enable = true;
services.openssh.settings.PermitRootLogin =
  "yes";
networking.firewall.enable = false;
```

</div>
<div data-bbox="419 579 738 600" data-label="Caption">
 Listing 4.1: NixOS Enable SSH
</div>
<div data-bbox="290 604 869 727" data-label="Text">
 Kemudian, kita akan menambahkan beberapa paket untuk mempermudah dalam melakukan penelitian. Kita akan menginstall neovim sebagai *text editor* dan tmux sebagai *terminal multiplexer* untuk menjaga sesi SSH apabila koneksi SSH ke *server* terputus.
</div>
<div data-bbox="290 731 810 810" data-label="Text">

```
environment.systemPackages = with pkgs; [
  neovim
  tmux
];
```

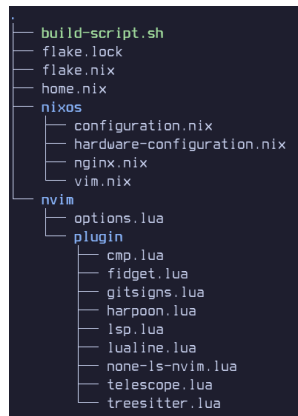
</div>
<div data-bbox="419 814 738 836" data-label="Caption">
 Listing 4.2: NixOS Enable SSH
</div>
<div data-bbox="290 842 869 903" data-label="Text">
 Untuk target, tambahkan opsi berikut agar nixos-rebuild dapat menginstall systemd-boot dengan versi lebih rendah dari yang telah terinstall.
</div>
<div data-bbox="513 936 547 955" data-label="Page-Footer">
 24
</div>



```
environment.sessionVariables =
{NIXOS_INSTALL_BOOTLOADER = "1"; };
```

*Listing 4.3: NixOS install bootloader*

2. Struktur Konfigurasi NixOS  
Untuk struktur konfigurasi yang digunakan adalah sebagai berikut:



*Gambar 4.6: Directory-tree*

Dalam konfigurasi ini, `flake.nix` dan `flake.lock` diletakkan di parent directory yang memang menjadi basis untuk dieksekusinya perintah `"nixos-rebuild switch --flake .#nixos-vm"`, dimana opsi `--flake` sendiri terdiri dari `flake-uri[#hostname]`

```
{
  inputs = {
    nixpkgs.url =
      "github:nixos/nixpkgs/nixos-23.11";
    home-manager = {
      url = "github:nix-community
/home-manager/release-23.11";
      inputs.nixpkgs.follows = "nixpkgs";
    };
    testing-website = {
```

```

        url =
        "github:rizqirazkafi/testing-website";
        flake = false;
    };
};
}

```

*Listing 4.4: flake inputs*

Flake.nix berisi tiga inputs yang dimana terdiri dari nixpkgs, home-manager, dan testing-website. Hasil dari inputs tadi digunakan dalam output untuk sistem yang didefinisikan dalam nixosConfiguration dengan nixos-vm sebagai hostname dari sistem.

```

[H]
{
    outputs = { self, nixpkgs, home-manager,
    ... }@inputs:
    let
        system = "x86_64-linux";
        pkgs = import nixpkgs {
            inherit system;
            config = { allowUnfree = true; };
        };
    in
    {
        nixosConfigurations = {
            nixos-vm = nixpkgs.lib.nixosSystem {
                specialArgs = { inherit inputs system pkgs; };
                modules = [ ./nixos/configuration.nix
                ];
            };
        };
    };
}

```

*Listing 4.5: flake outputs*

Dengan flake, versi paket yang dihasilkan dalam output tidak akan berubah baik dari channel maupun hash. Ini dikarenakan, informasi tersebut disimpan

dalam `flake.lock`. Dengan ini, sistem yang dihasilkan akan konsisten dan dapat disebut sebagai sistem yang *reproducible*.

Contoh konten `flake.lock` adalah sebagai berikut:

```
{
  "nodes": {
    "nixpkgs": {
      "locked": {
        "lastModified": 1708702655,
        "narHash": "sha256-qxT5jSxxx",
        "owner": "nixos",
        "repo": "nixpkgs",
        "rev":
          "c5101e457206dd437330d283d6626944e28794b3",
        "type": "github"
      },
      "original": {
        "owner": "nixos",
        "ref": "nixos-23.11",
        "repo": "nixpkgs",
        "type": "github"
      },
    },
  },
}
```

*Listing 4.6: nixpkgs lock di flake.lock*

3. Konfigurasi inti `configuration.nix`  
File `configuration.nix` berisikan konfigurasi sistem seperti *system packages*, *services*, *user packages*, *ssh option*, dan lain-lain. File ini bisa dibilang merupakan file inti dari konfigurasi sistem secara menyeluruh dimana file inilah yang di evaluasi oleh `nixos-rebuild`. Semua module yang dibutuhkan sistem harus di referensikan dalam file ini agar konfigurasi dapat diterapkan ke sistem. Ini merupakan sebagian konten file `configuration.nix` yang digunakan pada penelitian ini:

```

{inputs, pkgs, ... }: {
  nixpkgs.config.allowUnfree = true;
  environment.systemPackages = with pkgs; [
    lazygit
    tmux
    vim
    neovim
    wget
    git
    home-manager
    htop
    ranger
    ripgrep
    ansible
  ];
  security.pam.enableSSHAgentAuth = true;
  security.pam.enableSSHAgentAuth = true;
  services.openssh = {
    enable = true;
    ports = [ 9005 ];
    settings.PasswordAuthentication = false;
  };
}

```

#### 4. Home Manager

Untuk konfigurasi Home Manager, ada beberapa yang dimasukkan yaitu :

1. LSP server yang dibutuhkan untuk Neovim
2. plugin untuk Neovim
3. userName dan userEmail git
4. enableCompletion untuk BASH

Berikut merupakan sebagian dari konfigurasi yang dipakai untuk penelitian ini:

```

{config, pkgs, inputs, ...}:
{
  home.username = "rizqirazkafi";
  home.homeDirectory = "/home/rizqirazkafi";
  programs.neovim = let

```

```

toLua = str: ''
lua << EOF
${str}
EOF
'';
toLuaFile = file: ''
lua << EOF
${builtins.readFile file}
EOF
'';
in {
    enable = true;
    extraPackages = with pkgs; [
        # list of lsp
        xclip
        luajitPackages.lua-lsp
        rnix-lsp
        ansible-language-server
        nodePackages.bash-language-server
        shfmt
        ansible-lint
    ];
    plugins = with pkgs.vimPlugins; [
        vim-tmux-navigator
        nvim-lspconfig
        auto-pairs
        undotree
    ]
    {
        plugin = comment-nvim;
        config = toLua
            'require("Comment").setup()'';
    }
];
};
}

```

*Listing 4.7: Konfigurasi home.nix untuk Home-Manager*

## 5. Konfigurasi Ansible

Untuk Ansible menggunakan konfigurasi yang lebih sederhana dengan menyalin file konfigurasi dari vim, neovim, tmux, ssh, dan nginx ke *directory* konfigurasi sesuai Filesystem Hierarchical Standard (FHS). Untuk paket yang dibutuhkan di definisikan secara *global packages* dimana paket di install untuk semua *user*.

Untuk *Language Server Protocol* (LSP) *server* akan ditangani oleh Lazy.nvim dan Mason agar terinstall secara otomatis saat Neovim dijalankan.

Rincian konfigurasinya adalah sebagai berikut:

```
- hosts: all
  become: yes
  become_user: root
  tasks:
    - name: install packages
      ansible.builtin.apt:
        pkg:
          - hello
          - neovim
          - tmux
          - vim
          - wget
          - ....
```

*Listing 4.8: Konfigurasi instalasi paket di Ansible*

```
- name: copy nginx default config
  ansible.builtin.copy:
    src: default
    dest: /etc/nginx/sites-enabled/default
- name: copy php8.3-fpm config
  ansible.builtin.copy:
    src: php-demo.conf
    dest: /etc/php/8.3/fpm/pool.d
```

*Listing 4.9: Salin konfigurasi paket dan services ke target*

```
- name: enable nginx
  ansible.builtin.service:
    name: nginx
```

**state:** restarted

*Listing 4.10: Restart service menggunakan Ansible*

Maka dengan kedua *tool* Manajemen Konfigurasi inilah telah tercapai sistem GNU/Linux yang terdeklarasi dimana sistem terdeklarasikan dalam bentuk file-file konfigurasi.

## B. Pengujian

Tahap pengujian yang dilakukan dalam penelitian ini dimaksudkan untuk mengetahui waktu eksekusi yang dibutuhkan dari Ansible dan NixOS untuk menerapkan manajemen konfigurasi yang telah dibuat. Pengujian dilakukan dengan menggunakan perintah "*time*" pada eksekusi setiap kali dijalankan perintah untuk penerapan manajemen konfigurasi.

Pengujian berikutnya ialah menguji apakah baik Ansible maupun NixOS bersifat deklaratif dalam penerapannya menggunakan perbandingan dari apa yang telah di deklarasikan di konfigurasi dengan hasil akhir pada sistem.

### 1. Pasca Install

Pengujian dilakukan dengan menjalankan Ansible Playbook dan `nixos-rebuild` ke dua Virtual Machine yang baru saja di install. Tidak ada konfigurasi tambahan selain konfigurasi untuk IP menggunakan NetworkManager dan SSH. Masing-masing sistem juga berjalan di mode shell tanpa GUI. Pengujian ini akan dijalankan sebanyak tiga kali.

#### 1.a. NixOS tanpa cache

Pada percobaan ini, `nixos-rebuild` akan mengunduh semua *binary* dan *dependency* kemudian akan melakukan *build* berdasarkan *derivation* yang telah di definisikan oleh `nixpkgs`. Hasil dari *derivation* berupa *closure* yang disalin ke target.



Tabel 4.1: *nixos-rebuild first target*

real	user	sys
3m38.709s	0m12.379s	0m11.836s
3m34.648s	0m12.282s	0m11.082s
3m39.477s	0m12.199s	0m11.162s

Tabel 4.2: *nixos-rebuild second target*

real	user	sys
3m31.792s	0m12.334s	0m11.336s
3m33.855s	0m12.200s	0m11.512s
3m35.859s	0m12.278s	0m11.968s

## 1.b. NixOS dengan Cache

Pada percobaan ini, *nixos-rebuild* akan mengunduh semua *binary* dan *dependency* kemudian akan melakukan *build* berdasarkan *derivation* yang telah di definisikan oleh *nixpkgs*. Hasil dari *derivation* berupa *closure* yang disalin ke target.

Tabel 4.3: *nixos-rebuild first target*

real	user	sys
1m40.721s	0m3.563s	0m6.304s
1m39.842s	0m3.500s	0m6.616s
1m40.714s	0m3.463s	0m6.367s

Tabel 4.4: *nixos-rebuild second target*

real	user	sys
1m45.303s	0m3.544s	0m6.513s
1m43.955s	0m3.476s	0m6.326s
1m43.938s	0m3.523s	0m6.382s

### 1.c. Ansible Single Target

Pada percobaan ini, dijalankan menggunakan perintah ansible-playbook dengan satu target pada satu waktu. Total target yang di tuju adalah dua.

*Tabel 4.5: ansible first target*

real	user	sys
3m16.849s	0m8.208s	0m2.255s
2m45.881s	0m7.431s	0m2.095s
2m57.825s	0m7.761s	0m2.115s

*Tabel 4.6: ansible second target*

real	user	sys
2m48.054s	0m7.488s	0m2.119s
2m53.974s	0m7.598s	0m2.123s
2m.13.947s	0m7.341s	0m2.040s

### 1.d. Ansible Multi Target

Pada percobaan ini, dijalankan menggunakan perintah ansible-playbook dengan dua target secara paralel.

*Tabel 4.7: ansible multi target*

real	user	sys
2m12.327s	0m9.902s	0m3.188s
1m56.784s	0m9.654s	0m3.147s
2m2.801s	0m9.889s	0m2.932s

## 2. Penerapan ulang setelah pasca install.

Pengujian ini dilakukan dengan cara menjalankan Ansible Playbook dan nixos-rebuild setelah pengujian Pasca Install tanpa perubahan apapun pada file

konfigurasi. Pengujian ini akan dijalankan sebanyak tiga kali.

## 2.a. NixOS

*Tabel 4.8: nixos-rebuild first target*

<b>real</b>	<b>user</b>	<b>sys</b>
0m5.128s	0m0.210s	0m0.086s
0m4.186s	0m0.229s	0m0.073s
0m2.774s	0m0.228s	0m0.076s

*Tabel 4.9: nixos-rebuild second target*

<b>real</b>	<b>user</b>	<b>sys</b>
0m4.870s	0m0.222s	0m0.083s
0m4.222s	0m0.228s	0m0.071s
0m2.765s	0m0.234s	0m0.072s

## 2.b. Ansible single target

*Tabel 4.10: Ansible first target*

<b>real</b>	<b>user</b>	<b>sys</b>
0m16.252s	0m4.075s	0m0.997s
0m12.563s	0m3.931s	0m0.921s
0m13.220s	0m3.965s	0m0.975s

*Tabel 4.11: Ansible first target*

<b>real</b>	<b>user</b>	<b>sys</b>
0m13.381s	0m4.021s	0m0.976s
0m12.683s	0m3.924s	0m0.968s
0m12.958s	0m3.883s	0m0.951s

## 2.c. Ansible multi target

*Tabel 4.12: Ansible first target*

<b>real</b>	<b>user</b>	<b>sys</b>
0m14.552s	0m6.628s	0m1.886s
0m13.874s	0m6.402s	0m1.857s
0m14.230s	0m6.501s	0m1.818s

## 3. Penambahan Paket

Pengujian dilakukan dengan menambahkan paket tambahan yaitu Go compiler. Pengujian ini akan dilakukan tiga kali dengan satu kali perubahan pada file konfigurasi.

### 3.a. NixOS tanpa cache

*Tabel 4.13: nixos-rebuild first target*

<b>real</b>	<b>user</b>	<b>sys</b>
0m50.808s	0m6.168s	0m2.358s
0m46.404s	0m5.879s	0m1.860s
0m36.898s	0m5.832s	0m2.003s

*Tabel 4.14: nixos-rebuild second target*

<b>real</b>	<b>user</b>	<b>sys</b>
0m29.654s	0m5.908s	0m1.833s
0m41.280s	0m7.002s	0m2.446s
0m39.303s	0m6.856s	0m1.958s

### 3.b. NixOS dengan cache

*Tabel 4.15: nixos-rebuild first target*

<b>real</b>	<b>user</b>	<b>sys</b>
0m11.358s	0m0.483s	0m0.553s
0m10.317s	0m0.512s	0m0.477s
0m10.658s	0m0.486s	0m0.500s

*Tabel 4.16: nixos-rebuild second target*

<b>real</b>	<b>user</b>	<b>sys</b>
0m10.730s	0m0.498s	0m0.642s
0m9.745s	0m0.455s	0m0.474s
0m10.176s	0m0.488s	0m0.470s

### 3.c. Ansible single target

*Tabel 4.17: ansible first target*

<b>real</b>	<b>user</b>	<b>sys</b>
0m31.965s	0m4.378s	0m1.145s
0m23.455s	0m4.274s	0m1.025s
0m24.020s	0m4.283s	0m1.003s

*Tabel 4.18: ansible first target*

<b>real</b>	<b>user</b>	<b>sys</b>
0m26.637s	0m4.281s	0m1.120s
0m26.256s	0m4.327s	0m1.027s
0m25.256s	0m4.310s	0m1.046s

### 3.d. Ansible multi target

*Tabel 4.19: ansible multi target*

<b>real</b>	<b>user</b>	<b>sys</b>
0m33.164s	0m6.848s	0m2.088s
0m32.873s	0m7.054s	0m2.088s
0m36.413s	0m7.368s	0m2.129s

## 4. Penghapusan Paket

Pengujian dilakukan dengan menghapus paket yaitu Go compiler. Pengujian ini akan dilakukan tiga kali dengan satu kali perubahan pada file konfigurasi.

### 4.a. NixOS

*Tabel 4.20: nixos-rebuild first target*

<b>real</b>	<b>user</b>	<b>sys</b>
0m3.007s	0m0.219s	0m0.078s
0m2.918s	0m0.219s	0m0.085s
0m3.190s	0m0.218s	0m0.088s

*Tabel 4.21: nixos-rebuild second target*

<b>real</b>	<b>user</b>	<b>sys</b>
0m2.859s	0m0.218s	0m0.083s
0m2.582s	0m0.232s	0m0.067s
0m2.929s	0m0.219s	0m0.081s

#### 4.b. Ansible single target

*Tabel 4.22: ansible first target*

<b>real</b>	<b>user</b>	<b>sys</b>
0m15.627s	0m3.954s	0m0.988s
0m15.153s	0m3.928s	0m0.960s
0m15.504s	0m3.925s	0m0.979s

*Tabel 4.23: ansible second target*

<b>real</b>	<b>user</b>	<b>sys</b>
0m16.187s	0m4.021s	0m1.028s
0m15.441s	0m3.892s	0m1.019s
0m15.391s	0m3.925s	0m0.989s

#### 4.c. Ansible multi target

*Tabel 4.24: ansible multi target*

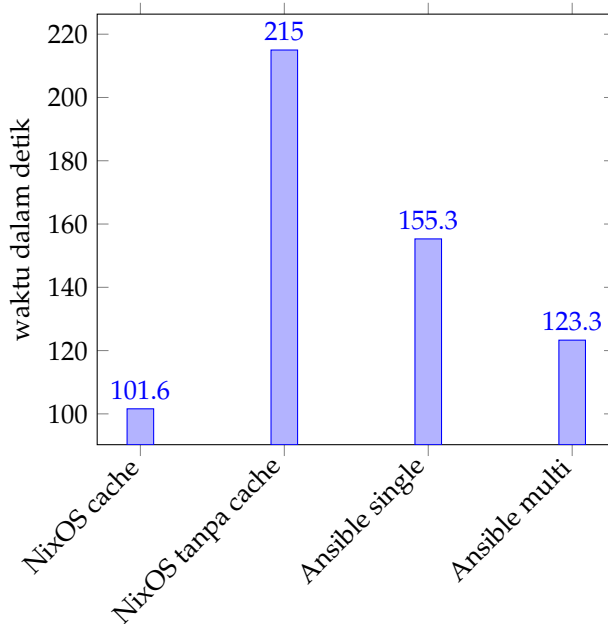
<b>real</b>	<b>user</b>	<b>sys</b>
0m17.768s	0m6.649s	0m1.986s
0m17.426s	0m6.628s	0m1.950s
0m18.449s	0m6.859s	0m2.088s

## 5. Grafik Perbandingan Waktu

Dari data-data *real time* diatas, maka didapatkan rata-rata data grafik waktu yang didapat dalam satuan detik adalah sebagai berikut:

### 5.a. Pasca Install

Dalam perhitungan ini, nixos cache, nixos tanpa cache, dan ansible single dihitung berdasarkan rata-rata dari 6 kali pengujian dari 2 target yang berbeda. Untuk ansible multi, diambil rata-rata dari 3 kali pengujian.

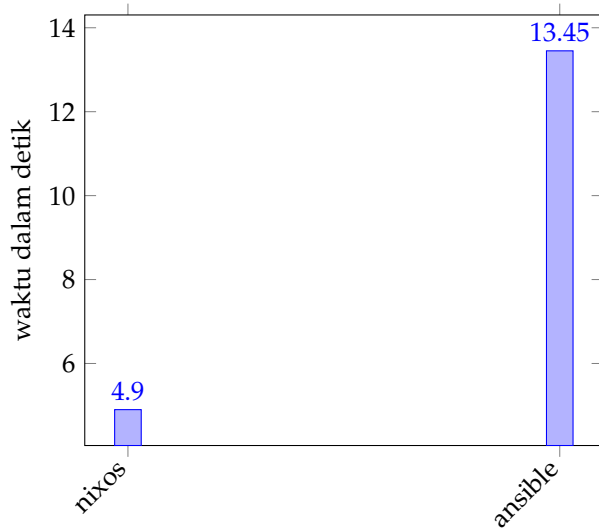


Dapat kita lihat bahwa NixOS tanpa cache butuh waktu lebih lama dikarenakan sistem *package manager* yang akan mengunduh *dependency* tiap paket secara terpisah apabila ada perbedaan versi. Faktor lain adalah tidak adanya sistem mirror seperti yang dimiliki oleh repository Ubuntu dimana paket bisa di simpan di server lain yang dekat, setidaknya tidak secara langsung. Namun apabila NixOS



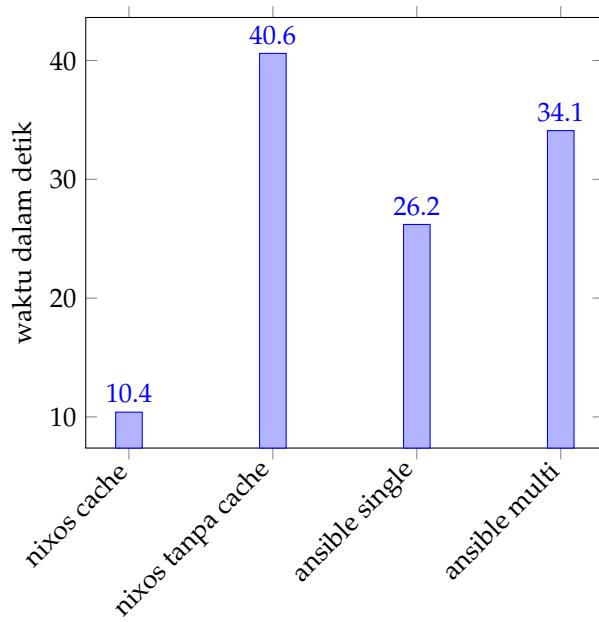
sudah menyimpan cache, maka NixOS hanya perlu menyalin data tersebut ke target dimana kecepatannya bergantung pada koneksi antara *host* dan target.

#### 5.b. Penerapan ulang setelah pasca install

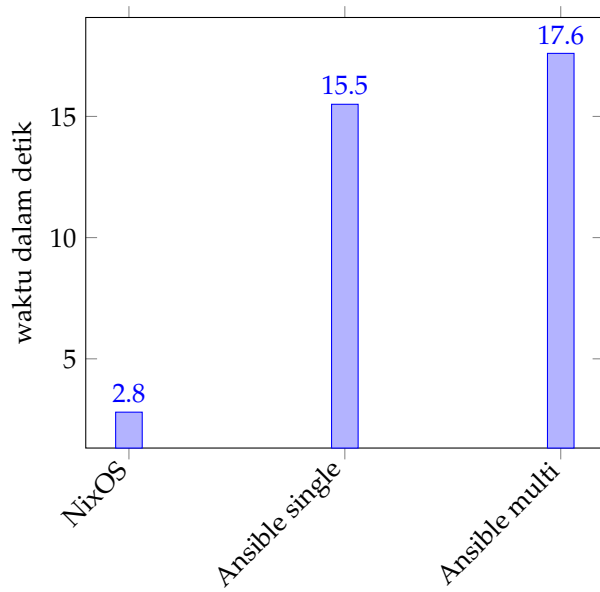


Disini dapat dilihat Ansible membutuhkan waktu lebih lama untuk melakukan pengecekan dikarenakan Ansible melakukan pengecekan tiap *task* pada setiap target sedangkan NixOS hanya mengecek perubahan pada konfigurasi dan membandingkan *closure* yang ada pada target dengan *host*.

### 5.c. Penambahan paket GO



#### 5.d. Penghapusan paket GO



## BAB V

### KESIMPULAN DAN SARAN

#### A. Kesimpulan

Berdasarkan penelitian perbandingan performa waktu eksekusi otomatisasi manajemen konfigurasi sistem operasi GNU/Linux antara Ansible dengan NixOS yang telah berhasil dilakukan, mendapatkan kesimpulan yang berdasarkan rumusan masalah, pembahasan, pengujian, dan hasil penelitian diantaranya:

1. Tujuan penulis adalah membuat sistem manajemen konfigurasi yang deklaratif dimana konfigurasi keseluruhan dari suatu sistem operasi diatur dengan *tool* manajemen konfigurasi. Dari penelitian ini, peneliti mendapati berbagai poin berikut:
  - a. Apabila task atau bagian dari Ansible Playbook dihapus, maka bagian tersebut akan tetap ada di sistem setelah Ansible Playbook dijalankan. Contohnya adalah ketika kita menghapus *package* atau *service*, maka *package* atau *service* tersebut akan tetap ada pada sistem. Apabila ingin menghapusnya, maka kita harus mendefinisikan di Ansible Playbook agar *package* atau *service* tersebut tidak ada dalam sistem. Ini membuktikan bahwa Ansible Playbook bersifat *imperatif* dimana Ansible Playbook hanya melakukan apa yang diperintahkan dan tidak memanifestasikan hasil akhir yang diinginkan.
  - b. NixOS dengan *nixos-rebuild* mampu memanifestasikan konfigurasi yang diterapkan pada sistem. Apabila *package* atau *service* dihapus dari konfigurasi, maka *package* atau *service* tersebut akan dihapus dari target.
2. Setelah melakukan pengujian dari waktu eksekusi kedua *tool* manajemen konfigurasi yaitu Ansible dan NixOS, hasilnya adalah sebagai berikut:

- a. Dalam uji coba pasca install, urutan waktu eksekusi dari yang tercepat hingga terlambat adalah NixOS dengan cache, Ansible *multu* target, Ansible single target, NixOS tanpa cache.
- b. Perintah `nixos-rebuild` tidak dapat digunakan untuk melakukan manajemen konfigurasi secara paralel sedangkan Ansible Playbook dapat dijalankan secara paralel. Ini berdampak pada penerapan dunia nyata dimana semakin banyak target maka waktu yang dibutuhkan untuk penerapan menggunakan `nixos-rebuild` akan memakan waktu lebih lama walaupun *cache / clojure* telah di *build*.

## B. Saran

Adapun saran yang dapat diberikan terkait hasil penelitian adalah sebagai berikut:

1. Disarankan membuat penelitian tentang perbandingan performa waktu eksekusi, maka bisa menggunakan Colmena atau NixOps agar bisa menerapkan konfigurasi NixOS secara paralel.
2. Disarankan bagi peneliti untuk mengubah jenis penyimpanan target dari HDD ke SSD untuk melihat apakah kecepatan penyimpanan target berpengaruh pada waktu eksekusi.

## DAFTAR PUSTAKA

- Alfiandi, T., Diansyah, T. M., & Liza, R. (2020). Analisis perbandingan manajemen konfigurasi menggunakan ansible dan shell script pada cloud server deployment aws. *JiTEKH*, 8, 78–84. <https://doi.org/10.35447/jitekh.v8i2.308>
- Ansible. (2016). Ansible is simple it automation. *Ansible.com*. Retrieved March 12, 2024, from <https://www.ansible.com/>
- Hariyadi, I. P., & Marzuki, K. (2020). Implementation of configuration management virtual private server using ansible. *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, 19, 347–357. <https://doi.org/10.30812/matrik.v19i2.724>
- Kumpulainen, K. (2019). Nixos: Järjestelmäkonfiguraation hallintaan erikoistunut linux-jakelu. *trepo.tuni.fi*. Retrieved March 17, 2024, from <https://urn.fi/URN:NBN:fi:tty-201905311795>
- NixOS. (2023). How nix works. *nixos.org*. Retrieved December 20, 2023, from <https://nixos.org/guides/how-nix-works/>
- Pratama, M. A. A., & Hariyadi, I. P. (2021). Otomasi manajemen dan pengawasan linux container (lxc) pada proxmox ve menggunakan ansible. *Jurnal Bumigora Information Technology (BITe)*, 3, 82–95. <https://doi.org/10.30812/bite.v3i1.807>
- Thufail Qolba, A. (2023, July). *Configuration management dengan ansible dan telegram untuk automasi laboratorium komputer di itik* [Doctoral dissertation]. Retrieved March 17, 2024, from <https://repository.pnj.ac.id/id/eprint/12406/1/Halaman%20Identitas%20Skripsi.pdf>