

**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

DAFTAR ISI

BAB I PENDAHULUAN	1
A. Latar Belakang	1
B. Rumusan Masalah	2
C. Tujuan	2
D. Manfaat	2
E. Batasan Masalah	3
BAB II KAJIAN PUSTAKA	4
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
BAB III METODE PENELITIAN	12
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	17
D.	Perancangan Pengujian	18
E.	Time Schedule	20
BAB IV	HASIL PENELITIAN DAN PEMBAHASAN	22
A.	Implementasi	22
1.	Struktur Konfigurasi NixOS	22
2.	Konfigurasi inti configuration.nix	24
3.	Home-Manager	25
DAFTAR PUSTAKA		26

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 alat 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 alat manajemen konfigurasi.

B. Rumusan Masalah

Adapun rumusan masalah berdasarkan latar belakang diatas yaitu:

1. Bagaimana membuat sistem GNU/Linux yang terdeklarasikan menggunakan alat manajemen konfigurasi.
2. Berapa lama waktu eksekusi alat 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 alat 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 alat 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 JTIC*" pada tahun 2023 (Thufail Qolba, 2023). Dalam penelitian tersebut menggunakan Ansible sebagai alat 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 terdeklarasikan 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 Alfiandi, 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 memperringkas pekerjaan dalam membangun *web server* (Alfiandi 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, Risiko 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

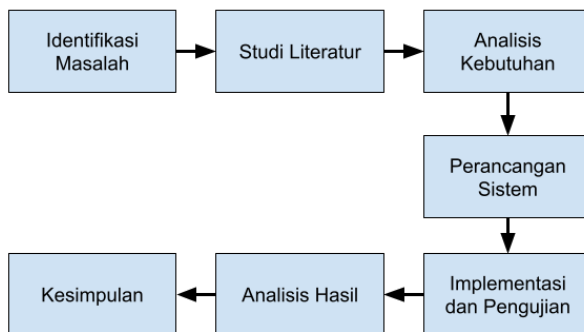
Virtualisasi merupakan proses berbasis perangkat lunak yang membagi komputer tunggal menjadi beberapa mesin virtual / virtual machine, tiap-tiap mesin virtual memiliki sistem operasi dan aplikasi miliknya sendiri (IBM, 2024)

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.



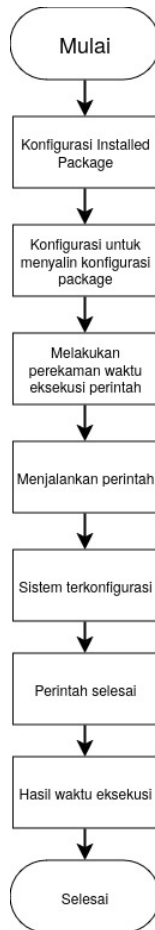
Gambar 3.1: Daigram experimental design

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 alat 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 alat 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 alat manajemen konfigurasi.

6. Analisis Hasil

Pada tahap ini, hasil dari implementasi dan pengujian akan dianalisis untuk mengetahui kecepatan penerapan konfigurasi dan konsistensi alat manajemen konfigurasi. Untuk mengetahui waktu yang dibutuhkan oleh alat 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 alat 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 23.11 (Tapir) 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	Debian	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 dua *virtual machine*, yang terdiri dari satu *instance* Debian 12 minimal sebagai host dan target Ansible. *Virtual machine* kedua merupakan guest kosong yang akan digunakan sebagai instalasi NixOS minimal.

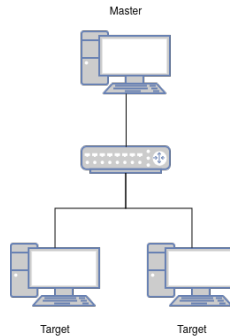
Tabel 3.2: Spesifikasi Instance

Instance Target Ansible	
Sistem Operasi	Debian 12
Processor	2
Memory	4 GB
Storage	50 GB
Jumlah <i>instance</i>	2
Instance Target NixOS	
Sistem Operasi	NixOS 23.11
Processor	2
Memory	4 GB
Storage	50 GB
Jumlah <i>instance</i>	2

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.



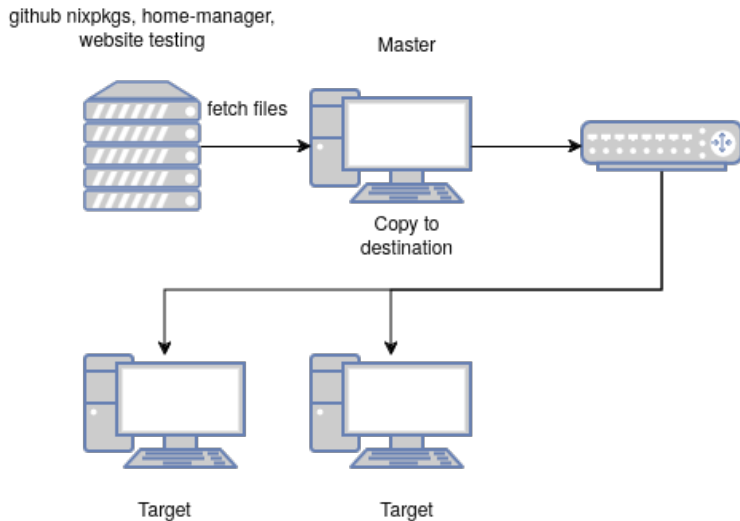
Gambar 3.3: Topologi Pengujian

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

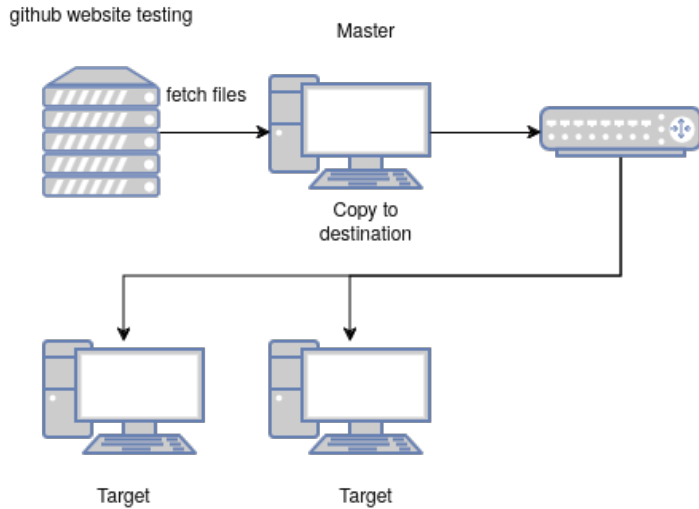
Perancangan pengujian pada penelitian ini akan melakukan instalasi berbagai macam perangkat lunak. Perangkat lunak yang diinstall adalah tmux, vim, neovim, http. Kemudian akan dilakukan beberapa konfigurasi perangkat lunak yaitu konfigurasi untuk tmux, vim, neovim, openssh, firewall.

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 3.3 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.

E. Time Schedule

Berikut *time schedule* yang penulis susun sebagai acuan pelaksanaan kegiatan:

Tabel 3.3: Time Schedule

Kegiatan	2023	2024				
	Des	Jan	Feb	Mar	Apr	Mei
Identifikasi Masalah						
Studi Literatur						
Analisis Kebutuhan						
Perancangan Sistem						
Implementasi dan Pengujian						
Analisis Hasil						
Kesimpulan						

BAB IV

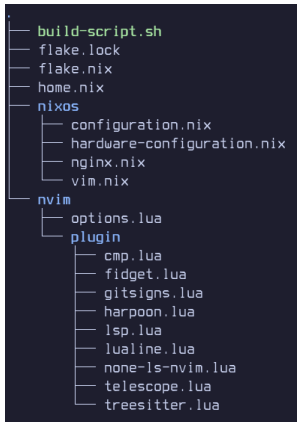
HASIL PENELITIAN DAN PEMBAHASAN

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



```
├── build-script.sh
├── flake.lock
├── flake.nix
├── home.nix
├── nixos
│   ├── configuration.nix
│   ├── hardware-configuration.nix
│   ├── nginx.nix
│   └── vim.nix
├── nvim
│   ├── options.lua
│   └── plugin
│       ├── cmp.lua
│       ├── fidget.lua
│       ├── gitsigns.lua
│       ├── harpoon.lua
│       ├── lsp.lua
│       ├── lua-line.lua
│       ├── none-ls-nvim.lua
│       ├── telescope.lua
│       └── treesitter.lua
```

Gambar 4.1: 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]`

```
1 description = "Home Manager configuration of rizqirazkafi";
2
3
4 inputs = {
5   # Specify the source of Home Manager and Nixpkgs.
6   nixpkgs.url = "github:nixos/nixpkgs/nixos-23.11";
7   home-manager = {
8     url = "github:nix-community/home-manager/release-23.11";
9     inputs.nixpkgs.follows = "nixpkgs";
10  };
11  testing-website = {
12    url = "github:rizqirazkafi/testing-website";
13    flake = false;
14  };
15 };
16
17 outputs = { self, nixpkgs, home-manager, ... }@inputs:
18   let
19     system = "x86_64-linux";
20     pkgs = import nixpkgs {
21       inherit system;
22       config = { allowInfinite = true; };
23     };
24   in
25   {
26     nixosConfigurations = {
27       nixos-vm = nixpkgs.lib.nixosSystem {
28         specialArgs = { inherit inputs system pkgs; };
29         modules = [ ./nixos/configuration.nix ];
30       };
31     };
32  };
```

Gambar 4.2: *flake.nix*

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.

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:

```

"nixpkgs": {
  "locked": {
    "lastModified": 1708702655,
    "narHash": "sha256-qxT5jSLhe1fLhQ07+AUxSTm1VnVH+hQxDkQSZ/m/Smo=",
    "owner": "nixos",
    "repo": "nixpkgs",
    "rev": "c5101e457206dd437330d283d6626944e28794b3",
    "type": "github"
  },
  "original": {
    "owner": "nixos",
    "ref": "nixos-23.11",
    "repo": "nixpkgs",
    "type": "github"
  }
}

```

Gambar 4.3: nixpkgs

2. 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:

```

# List packages installed in system profile. To search, run:
# $ nix search wget
nixpkgs.config.allowUnfree = true;
environment.systemPackages = with pkgs; [
  lazygit
  tmux
  vim
  neovim
  wget
  git
  home-manager
  htop
  ranger
  ripgrep
  ansible
];

```

Gambar 4.4: Definisi paket dalam configuration.nix

```

security.pam.enableSSHAgentAuth = true;
services.openssh = {
  enable = true;
  banner = ''
    -----
    \_//  _//  _//  _//  _//  _//  _//  _//
    \_//  _//  _//  _//  _//  _//  _//  _//
    -----
    //  _//  _//  _//  _//  _//  _//  _//
    //  _//  _//  _//  _//  _//  _//  _//
    -----
  '';
  ports = [ 9005 ];
  settings.PasswordAuthentication = false;
};

```

Gambar 4.5: Definisi opsi ssh configuration.nix

```

# Use the systemd-boot
boot.loader.systemd-boot.enable = true;
boot.loader.systemd-boot.configurationLimit = 10;
boot.loader.efi.canTouchEfiVariables = true;

networking.hostName = "nixos-vm"; # Define your hostname.
networking.nameservers = [ "1.1.1.1" "8.8.8.8" ];
networking.networkmanager.enable = true;

```

Gambar 4.6: Definisi opsi bootloader configuration.nix

3. Home-Manager

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 (lcx) 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 jtik* [Doctoral dissertation]. Retrieved March 17, 2024, from <https://repository.pnj.ac.id/id/eprint/12406/1/Halaman%20Identitas%20Skripsi.pdf>