

**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**

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

SKRIPSI

Diajukan kepada Universitas Negeri Surabaya
untuk memenuhi persyaratan penyelesaian
Program Sarjana Komputer

Oleh
M. RIZQI R
NIM 20051204034

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

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. Pada sisi lain, Ansible terbukti lebih cepat dan efektif apabila mengkonfigurasi sistem lebih dari satu karena Ansible memiliki kemampuan paralelisasi sedangkan NixOS dengan nixos-rebuild belum mampu melakukan paralelisasi.

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

ABSTRACT

OPERATING SYSTEM CONFIGURATION MANAGEMENT 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 the goal of facilitating the configuration management process and ensure the desired end result will be the same every time execution. As a configuration management tool, Ansible has a faster execution time than NixOS with nixos-rebuild. However, from a declarative point of view, NixOS proved to be to be more declarative because everything in the configuration is manifested in the system. Unlike Ansible which only does what is in the Ansible Playbook and does not manifest in the system. and does not become a manifestation of the system. On the other hand, Ansible proves to be faster and more effective when configuring more than one system because Ansible has the ability to configure more than one system because Ansible

has the ability to parallelize parallelization capabilities whereas NixOS with nixos-rebuild is not yet capable of parallelization.

Keywords - Ansible, NixOS, *declarative*, *imperative*, Configuration Management

SURAT PERNYATAAN KEORISINILAN SKRIPSI

Yang bertandatangan dibawah ini:

Nama Mahasiswa : M. Rizqi R
Tempat, Tanggal Lahir : Sidoarjo, 31 Juli 2001
NIM : 20051204034
Program Studi / Angkatan : S1 Teknik Informatika
Alamat : WADUNGASIH, RT/RW
005/002 DESA
WADUNGASIH,
KECAMATAN BUDURAN,
KABUPATEN SIDOARJO

Menyatakan dengan sesungguhnya bahwa:

1. Skripsi yang ditulis dan diajukan ini benar-benar merupakan hasil karya saya sendiri (tidak berdiri berdasarkan pada data palsu, dan atau hasil plagiasi atau jiplakan atau autoplagiasi).
2. Apabila pada kemudian hari terbukti bahwa pernyataan saya tidak benar, maka saya akan menanggung segala resiko dan siap diperkarakan sesuai aturan yang berlaku.

Demikianlah surat pernyataan yang saya buat dengan sebenar-benarnya.

Surabaya, 11 Juni 2024

M. Rizqi R
NIM 20051204034

HALAMAN PERSETUJUAN SKRIPSI



KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN RISET, DAN TEKNOLOGI
UNIVERSITAS NEGERI SURABAYA
FAKULTAS TEKNIK
JURUSAN TEKNIK INFORMATIKA
Kampus Ketintang, Jalan Ketintang, Surabaya 60231
Telepon: +6231-8280009 pes.500 - 510, Faksimil: +6231-8280796
Laman: if.unesa.ac.id, email: if@unesa.ac.id

HALAMAN PERSETUJUAN SKRIPSI

Usulan Skripsi oleh

1. NIM : M. Rizqi R
2. NAMA : 20051204060
3. Program Studi : S1 Teknik Informatika
4. Semester : 8, Genap 2023/2024
5. Judul Artikel Ilmiah : PERBANDINGAN WAKTU EKSEKUSI OTOMASI MANAJEMEN KONFIGURASI SISTEM OPERASI GNU/LINUX ANTARA ANSIBLE DENGAN NIXOS

Ini telah disetujui dan dinyatakan memenuhi syarat untuk dilakukan penilaian.

Surabaya, 20 Mei, 2024
Dosen Pembimbing

Agus Prihanto S.T., M.Kom
NIP. 197908062006041001

HALAMAN PENGESAHAN

Skripsi Oleh : M. Rizqi R
NIM : 20051204034
Judul : Perbandingan Waktu Eksekusi Otomasi
Manajemen Konfigurasi Sistem Operasi
GNU/Linux Antara Ansible dengan NixOS

Ini telah dipertahankan dihadapan dewan penguji pada tanggal
6 Juni 2024

HALAMAN PERSEMBAHAN

Skripsi ini saya tujuhan dan persembahkan kepada:

1. Allah SWT, oleh karena rahmat dan hidayah-Nya, serta segala kenikamatan dunia sehingga skripsi ini dapat ditulis dan disusun dengan baik dan selesai pada waktunya.
2. Orang tua terkasih, Bapak Wahyu Diarto dan Ibu Elmi Solichah yang selalu mendukung peneliti dalam segala aspek terutama menempuh pendidikan.
3. Adik-adik peneliti, yaitu M. Hilman AS dan Anis Wahyu yang secara tidak langsung memberi hiburan kepada peneliti ketika pulang ke rumah.
4. Teman-teman Asisten Laboratorium Jaringan Komputer dan Asisten Laboratorium Sistem Informasi, terutama Dani Maulana Ferdiansyah, Jerry Yan Krismanto, Firdaus Bagus Wicaksono, serta anggota Aslab lain yang membantu peneliti baik dalam proses penelitian maupun pemberkasan.
5. Rekan selingkuh Teknik Informatika, khususnya Ahmad Fikry Husaini, Dito Arya Saputra, Mochammad Subhan Zuliananta, Ricardo Sinulingga yang telah menemani peneliti bermain game saat jenuh.
6. Vimjoyer dan IogaMaster atas channel YouTube dan forum Discord yang telah membantu dalam mempelajari NixOS
7. Diri sendiri, yang selalu mencoba untuk bangkit kembali dari kemalasan untuk mengerjakan skripsi sehingga dapat lulus tepat waktu dan tidak membayar UKT untuk penambahan semester.

DAFTAR ISI

ABSTRAK	ii
ABSTRACT	iv
DAFTAR TABEL	xii
DAFTAR GAMBAR	xiii
BAB I PENDAHULUAN	1
A Latar Belakang	1
B Rumusan Masalah	2
C Tujuan	2
D Manfaat	3
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	18
D	Perancangan Pengujian	18
BAB IV HASIL PENELITIAN DAN PEMBAHASAN		21
A	Implementasi	21
1	Persiapan Lingkungan NixOS	21
2	Struktur Konfigurasi NixOS	25
3	Konfigurasi inti configuration.nix	27
4	Home Manager	28
5	Persiapan Lingkungan Ansible	29
6	Konfigurasi Ansible	31
B	Pengujian	33
1	Pasca Install	33
2	Penerapan ulang setelah pasca install.	40
3	Penambahan Paket	45
4	Penghapusan Paket	54
5	Grafik Perbandingan Waktu	60
6	Hasil Konfigurasi	64
7	Deklaratif dan Imperatif	65
BAB V KESIMPULAN DAN SARAN		69
A	Kesimpulan	69
B	Saran	70

DAFTAR PUSTAKA	70
-----------------------------	-----------

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 tanpa cache	35
Tabel 4.2 nixos-rebuild second target tanpa cache	35
Tabel 4.3 nixos-rebuild first target dengan cache	37
Tabel 4.4 nixos-rebuild second target	37
Tabel 4.5 Ansible first target pasca install	39
Tabel 4.6 ansible second target pasca install	39
Tabel 4.7 ansible multi target pasca install	40
Tabel 4.8 nixos-rebuild first target pasca install rerun	43
Tabel 4.9 nixos-rebuild second target pasca install rerun	43
Tabel 4.10 Ansible first target rerun	44
Tabel 4.11 Ansible second target rerun	44
Tabel 4.12 Ansible multi target	45
Tabel 4.13 nixos-rebuild first target	47
Tabel 4.14 nixos-rebuild second target	48
Tabel 4.15 nixos-rebuild first target install Go cache	51
Tabel 4.16 nixos-rebuild second target install Go cache	51
Tabel 4.17 ansible first target	52
Tabel 4.18 ansible second target	53
Tabel 4.19 Ansible multi target install Go	54
Tabel 4.20 nixos-rebuild first target uninstall Go	57
Tabel 4.21 nixos-rebuild second target uninstall Go	57
Tabel 4.22 ansible first target	59
Tabel 4.23 ansible second target	59
Tabel 4.24 ansible multi target	60

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	19
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
4.7	Ubuntu 24.04 Live Server	29
4.8	Repository Ubuntu	30
4.9	Ubuntu Server Disk	30
4.11	Ubuntu Server Telah terinstall	31
4.10	Ubuntu Server Disk Layout	31
4.12	nixos-rebuild first target tanpa cache	34
4.13	nixos-rebuild second target tanpa cache	35
4.14	nixos-rebuild first target dengan cache	36
4.15	nixos-rebuild second target dengan cache	37
4.16	Ansible first target pasca install	38
4.17	Ansible second target pasca install	39
4.18	Ansible multi target pasca install	40
4.19	nixos-rebuild first target pasca install rerun	41
4.20	nixos-rebuild second target pasca install rerun	42
4.21	Ansible first target rerun	43
4.22	Ansible second target rerun	44
4.23	Ansible multi target rerun	45

4.24 nixos-rebuild first target install Go	46
4.25 nixos-rebuild second target install Go	47
4.26 nixos-rebuild first target install Go cache	49
4.27 nixos-rebuild second target install Go cache	50
4.28 Ansible first target install Go	52
4.29 Ansible second target install Go	52
4.30 Ansible multi target install Go	53
4.31 nixos-rebuild first target uninstall Go	55
4.32 nixos-rebuild second target uninstall Go	56
4.33 Ansible first target uninstall Go	58
4.34 Ansible second target uninstall Go	58
4.35 Ansible multi target uninstall Go	60
4.36 Hasil konfigurasi NixOS	65
4.37 Hasil konfigurasi Ansible	65
4.38 Nix Nginx enable	66
4.39 Nix Nginx removed	67
4.40 Enable nginx	67
4.41 Nginx masih berjalan	68

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 operasi 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 dengan hasil akhir konfigurasi yang serupa.

B. Rumusan Masalah

Adapun rumusan masalah berdasarkan latar belakang diatas yaitu:

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

C. Tujuan

Adapun tujuan dalam penelitian ini adalah:

1. Membuat sistem GNU/Linux yang terdeklarasi menggunakan NixOS dan Ansible dengan target akhir hasil konfigurasi yang serupa.
2. Mengukur waktu eksekusi *tool* manajemen konfigurasi NixOS menggunakan nixos-rebuild dan Ubuntu menggunakan 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 dimana hasil akhirnya serupa.
4. Perbandingan akan dilihat berdasarkan waktu yang dibutuhkan untuk eksekusi penerapan konfigurasi.
5. Untuk nixos-rebuild hanya dijalankan di target NixOS dan Ansible hanya akan dijalankan di Ubuntu menggunakan *apt package manager*.

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 JTIK*" 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 Alfiandi, T.M Diansyah, dan Risko 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 memperingkas 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 dengan Ansible dan Telegram Untuk Automasi Laboratorium Komputer di JTIK</i>	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 Erikoinen 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

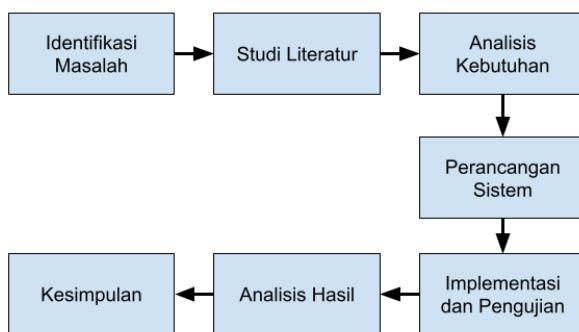
Virtualisasi merupakan proses berbasis perangkat lunak yang membagi komputer tunggal menjadi beberapa mesin virtual / virtual machine, tiap-tiap mesin virtual memiliki sistem operasi dan aplikasi miliknya sendiri (IBM, 2024)

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

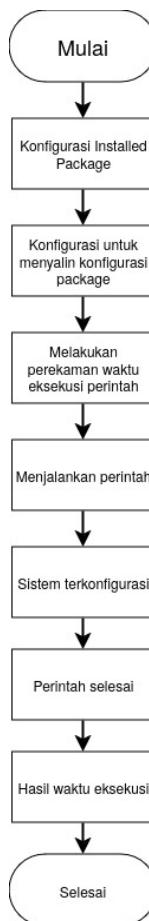
Peneliti mencari literatur yang relevan untuk menjadi referensi pendukung dalam penelitian. Literatur yang digunakan dalam penelitian ini berhubungan dengan penerapan manajemen konfigurasi yang didapat dari berbagai macam sumber seperti buku, artikel, jurnal nasional maupun internasional.

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 dari hasil akhir konfigurasi setelah diterapkan secara berulang.

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

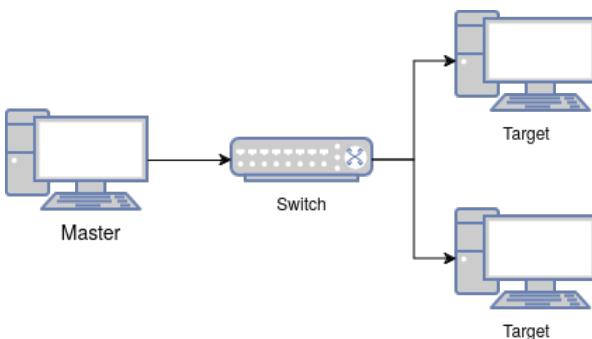
<i>Instance Ansible</i>	
Sistem Operasi	Ubuntu Server 24.04 LTS
Processor	2
Memory	4 GB
Storage	50 GB
Jumlah <i>instance</i>	3
<i>Instance NixOS</i>	
Sistem Operasi	NixOS 23.11
Processor	2
Memory	4 GB
Storage	50 GB
Jumlah <i>instance</i>	3

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. Alasan berikutnya adalah dikarenakan kenyataan

dilapangan dimana Ubuntu Server akan dikonfigurasi menggunakan Ansible dan menggunakan *package manager* apt, sedangkan untuk NixOS pasti akan dikonfigurasi menggunakan file konfigurasi ".nix".

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

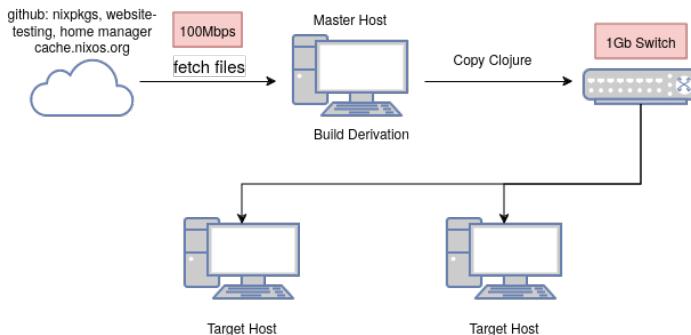
Dengan topologi ini, diharapkan penerapan manajemen konfigurasi terpusat dan dapat diterapkan di dua mesin target dan memiliki hasil akhir yang sama dan konsisten antar mesin.

D. Perancangan Pengujian

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

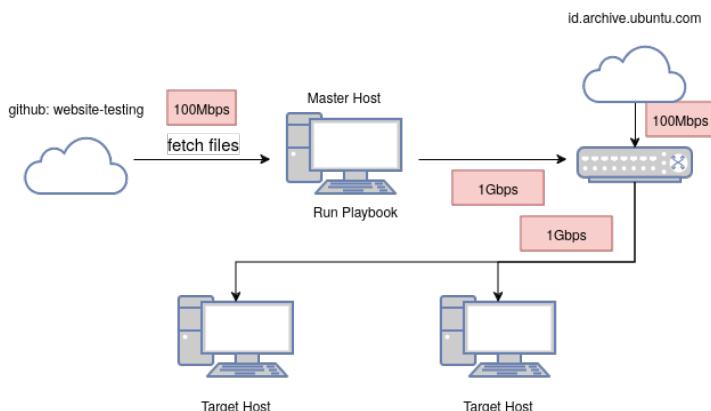
Kemudian akan dihadirkan service tambahan yaitu berupa *web server* menggunakan Nginx. Nginx juga akan

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. Masing-masing target dengan sistem operasi Ubuntu Server akan mengunduh paket dari *repository* id.archive.ubuntu.com. 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. Patut di ingat bahwa *bandwidth* maksimal untuk menuju internet adalah 100Mbps dan bandwidth *local connection* adalah 1Gbps.

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

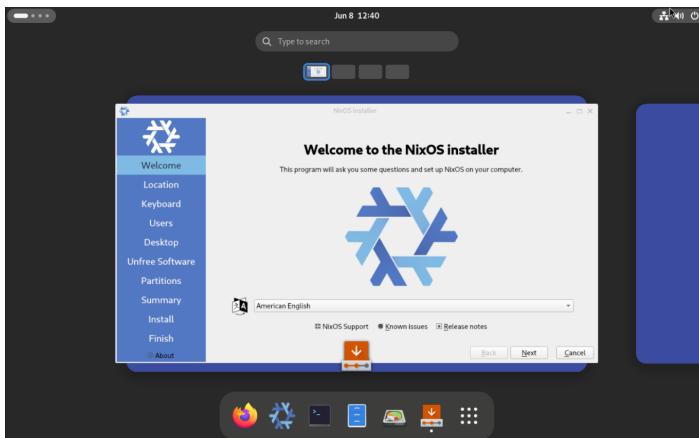
1. Persiapan Lingkungan NixOS

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 *firmware setting* 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:

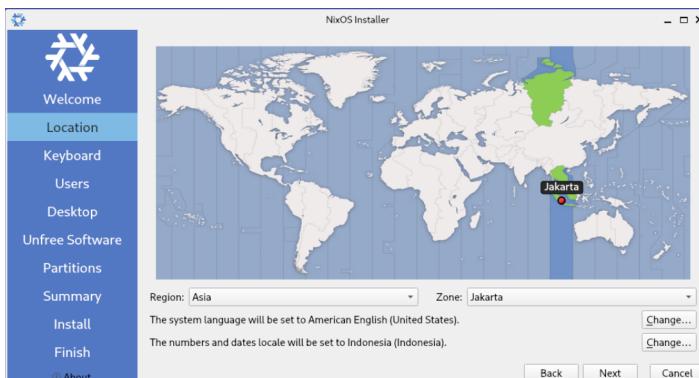


Gambar 4.1: NixOS 23.11 installer GNOME

1.b. Instalasi

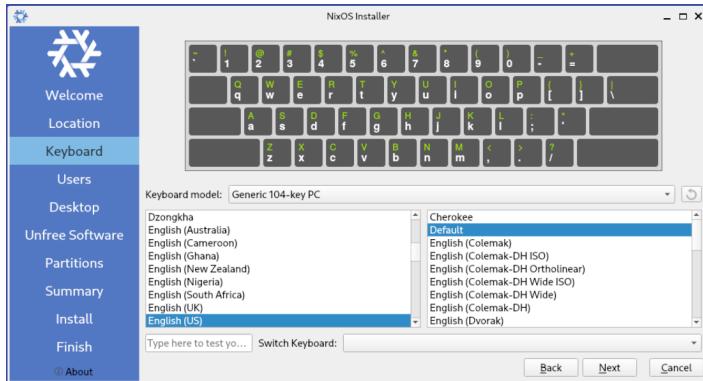
Kemudian kita masuk ke tahap instalasi NixOS sebagai berikut:

1. Atur lokasi ke Asia/Jakarta.



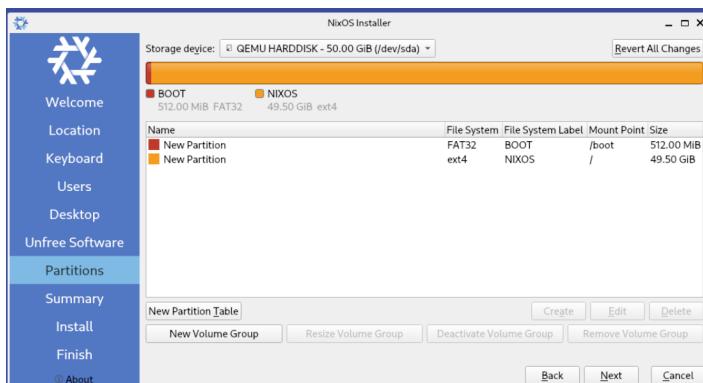
Gambar 4.2: NixOS 23.11 Location

2. Atur keyboard ke US default.



Gambar 4.3: NixOS 23.11 Keyboard

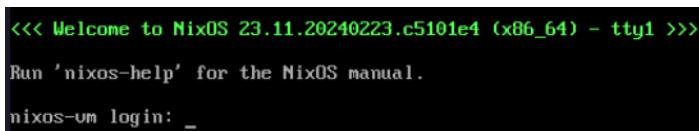
- Buat username dan password kemudian centang bagian "Use the same password for the administrator account".
 - Untuk pilihan Desktop, pilih "No desktop" karena kita hanya butuh akses SSH.
 - Centang bagian "Allow Unfree Software".
 - Buat skema partisi seperti berikut dengan tabel partisi GPT:



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:



```
<<< Welcome to NixOS 23.11.20240223.c5101e4 (x86_64) - tty1 >>>
Run 'nixos-help' for the NixOS manual.

nixos-vm login: _
```

Gambar 4.5: NixOS 23.11 tty

1.c. Setup OS

Setelah itu, kita aktifkan SSH dan mematikan *firewall* dengan menambahkan baris dibawah ini kedalam file /etc/nixos/configuration.nix:

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

Listing 4.1: NixOS Enable SSH

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.

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

Listing 4.2: NixOS Enable SSH

Untuk target, tambahkan opsi berikut agar nixos-rebuild dapat menginstall systemd-boot dengan versi lebih rendah dari yang telah terinstall.

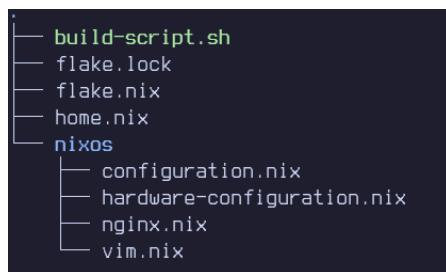
```
environment.sessionVariables =
{NIXOS_INSTALL_BOOTLOADER = "1"; };
```

Listing 4.3: NixOS install bootloader

Setelah itu bisa diterapkan dengan perintah “`sudo nixos-rebuild switch`”. Apabila tidak di spesifikasikan *path*, maka secara *default* akan mengarah ke `/etc/nixos/configuration.nix`.

2. Struktur Konfigurasi NixOS

Untuk struktur konfigurasi target 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 berikan 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. userName dan userEmail git
2. enableCompletion untuk BASH

Berikut merupakan sebagian dari konfigurasi yang dipakai untuk penelitian ini:

```

{config, pkgs, inputs, ...}:
{
  home.username = "rizqirazkafi";
  home.homeDirectory = "/home/rizqirazkafi";
  programs.git = {
    enable = true;
    userName = "Rizqirazkafi";
    userEmail = "rizqir*****@gmail.com";
  };
}

```

```

programs.bash = {
  enable = true;
  shellAliases = { ll = "ls -la"; };
  enableCompletion = true;
  initExtra = ''echo "Hello, what good
shall I do today?'''';
};

}

```

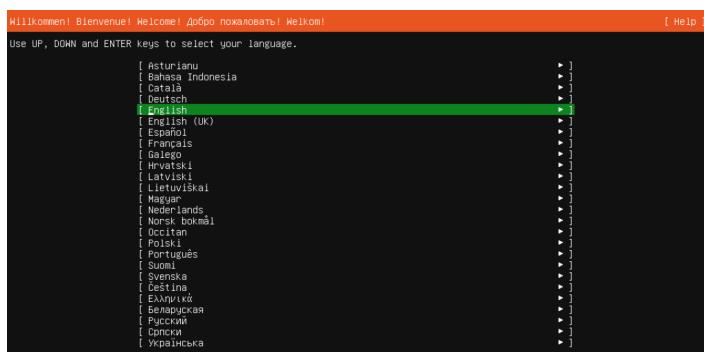
Listing 4.7: Konfigurasi home.nix untuk Home-Manager

5. Persiapan Lingkungan Ansible

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

5.a. Download ISO dan konfigurasi VM

Download ISO Ubuntu 24.04 live server di <https://ubuntu.com> atau <https://kartolo.sby.datautama.net.id/ubuntu-cd>. Buat VM di Proxmox dengan konfigurasi yang sama dengan Lingkungan NixOS. Karena Ubuntu mendukung *secure boot* sepenuhnya, kita tidak perlu mematikan *secure boot* pada *firmware setting* seperti NixOS. Ketika sudah *booting*, maka akan muncul tampilan sebagai berikut:

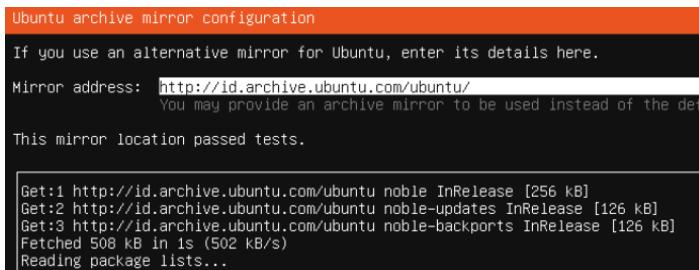


Gambar 4.7: Ubuntu 24.04 Live Server

5.b. Instalasi

Kemudian kita masuk ke tahap instalasi Ubuntu Server sebagai berikut:

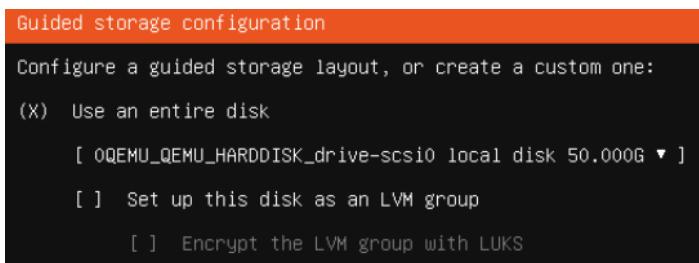
1. Pilih bahasa dan keyboard yang diinginkan, penulis memilih Bahasa Inggris dan *layout* US.
2. Pilih basis instalasi Ubuntu Server.
3. Untuk IP, penulis menggunakan yang diberikan oleh DHCP server.
4. Kosongkan proxy.
5. Untuk *mirror address* penulis menggunakan <https://id.archive.ubuntu.com/ubuntu>



Gambar 4.8: Repository Ubuntu

Pastikan *repository* dapat dijangkau.

6. Untuk *disk* kita gunakan seluruhnya, untuk LVM dimatikan karena tidak dibutuhkan.



Gambar 4.9: Ubuntu Server Disk

Hasilnya harusnya seperti ini:

```
Ubuntu 24.04 LTS ansible-nix-1 tty1

ansible-nix-1 login: [ 24.793898] cloud-init[1053]: Cloud-init v. 24.0.1
[...]
ci-info: no authorized SSH keys fingerprints found for user rizqirazka
<14>Jun 10 04:07:32 cloud-init: ##### BEGIN SSH HOST KEY FINGERPRINTS #####
<14>Jun 10 04:07:32 cloud-init: -----BEGIN SSH HOST KEY FINGERPRINTS-----
<14>Jun 10 04:07:32 cloud-init: 256 SHA256:/rQgKTSFNx+3KD9jLva6xUqlK8Y
<14>Jun 10 04:07:32 cloud-init: 256 SHA256:WleusPKKdmncGSpvrm7gvwz6Gx0
<14>Jun 10 04:07:32 cloud-init: 3072 SHA256:tX0kztosDXdee4+H27818CMv10
<14>Jun 10 04:07:32 cloud-init: -----END SSH HOST KEY FINGERPRINTS-----
<14>Jun 10 04:07:32 cloud-init: ##### BEGIN SSH HOST KEY KEYS #####
<14>Jun 10 04:07:32 cloud-init: ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAA
root@ansible-nix-1:~$
```

Gambar 4.11: Ubuntu Server Telah terinstall

FILE SYSTEM SUMMARY					
MOUNT POINT	SIZE	TYPE	DEVICE	TYPE	
/	48.948G	new ext4	new partition of	local disk	►
[/boot/efi	1.049G	new fat32	new partition of	local disk	►

Gambar 4.10: Ubuntu Server Disk Layout

7. Kita skip Ubuntu Pro
8. Kita tidak akan install SSH server pada saat instalasi
9. Lanjut proses instalasi
10. Apabila sudah selesai dan Ubuntu Server berhasil terinstall, maka hasilnya akan seperti ini:

6. Konfigurasi Ansible

Untuk Ansible menggunakan konfigurasi yang lebih sederhana dengan menyalin file konfigurasi dari vim, tmux, ssh, dan nginx ke *directory* konfigurasi sesuai Filesystem Hierachial 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, *master* 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*. Karena nixos-rebuild tidak dapat berjalan secara paralel, maka percobaan dilakukan secara bergantian untuk masing-masing target. Setelah nixos-rebuild menyalin *closure* ke *target*, maka waktu eksekusi telah didapat. Setelah waktu eksekusi didapat, baik VM untuk *target* dan *master* akan dikembalikan ke *snapshot* sebelum dijalankannya nixos-rebuild. Percobaan dilakukan tiga kali untuk masing-masing target.

```
reloading the following units: dbus.service, reload-systemd-vconsole-setup.service
restarting the following units: boot.mount, dev-hugepages.mount, sshd.service
starting the following units: NetworkManager-wait-online.service, NetworkManager.service, network-local-commands.service, network-setup.service, nscd.service, resolvectl.socket, systemd-sysctl.service, systemd-timesyncd.service, systemd-tmpfiles-setup.service
the following new units were started: NetworkManager-dispatcher.service, firewalld.service, phpfm.slice, phpfm.target, systemd-fsck@dev-disk-by\x2dlabel-800T.service

real    3m29.511s
user    0m12.695s
sys     0m14.709s

reloading the following units: dbus.service, reload-systemd-vconsole-setup.service
restarting the following units: boot.mount, dev-hugepages.mount, sshd.service, sysctl.service
starting the following units: NetworkManager-wait-online.service, NetworkManager.service, network-local-commands.service, network-setup.service, nscd.service, resolvectl.socket, systemd-sysctl.service, systemd-timesyncd.service, systemd-tmpfiles-setup.service
the following new units were started: NetworkManager-dispatcher.service, firewalld.service, phpfm.slice, phpfm.target, systemd-fsck@dev-disk-by\x2dlabel-800T.service

real    3m27.362s
user    0m12.978s
sys     0m14.277s

reloading the following units: dbus.service, reload-systemd-vconsole-setup.service
restarting the following units: boot.mount, dev-hugepages.mount, sshd.service, sysctl.service
starting the following units: NetworkManager-wait-online.service, NetworkManager.service, network-local-commands.service, network-setup.service, nscd.service, resolvectl.socket, systemd-sysctl.service, systemd-timesyncd.service, systemd-tmpfiles-setup.service
the following new units were started: NetworkManager-dispatcher.service, firewalld.service, phpfm.slice, phpfm.target, systemd-fsck@dev-disk-by\x2dlabel-800T.service

real    3m28.368s
user    0m12.687s
sys     0m14.586s
```

Gambar 4.12: nixos-rebuild first target tanpa cache

```

reloading the following units: dbus.service, reload-systemd-vconsole-setup.service
restarting the following units: boot.mount, dev-hugepages.mount, nix-daemon.service
starting the following units: NetworkManager-wait-online.service, NetworkManager.service,
network-local-commands.service, network-setup.service, nscd.service, resolv-dns-socket,
systemd-sysctl.service, systemd-timesyncd.service, systemd-tmpfiles-
e-done.service
the following new units were started: NetworkManager-dispatcher.service, firewall.,
phpfpm.target, systemd-fsck@dev-disk-by\x2dlabel-800T.service, systemd-hostnamed

real    3m28.804s
user    0m12.822s
sys     0m14.634s

reloading the following units: dbus.service, reload-systemd-vconsole-setup.service
restarting the following units: boot.mount, dev-hugepages.mount, nix-daemon.service
starting the following units: NetworkManager-wait-online.service, NetworkManager.service,
network-local-commands.service, network-setup.service, nscd.service, resolv-dns-socket,
systemd-sysctl.service, systemd-timesyncd.service, systemd-tmpfiles-
e-done.service
the following new units were started: NetworkManager-dispatcher.service, firewall.,
phpfpm.target, systemd-fsck@dev-disk-by\x2dlabel-800T.service, systemd-hostnamed

real    3m27.731s
user    0m12.900s
sys     0m14.437s

reloading the following units: dbus.service, reload-systemd-vconsole-setup.service
restarting the following units: boot.mount, dev-hugepages.mount, nix-daemon.service
starting the following units: NetworkManager-wait-online.service, NetworkManager.service,
network-local-commands.service, network-setup.service, nscd.service, resolv-dns-socket,
systemd-sysctl.service, systemd-timesyncd.service, systemd-tmpfiles-
e-done.service
the following new units were started: NetworkManager-dispatcher.service, firewall.,
phpfpm.target, systemd-fsck@dev-disk-by\x2dlabel-800T.service, systemd-hostnamed

real    3m36.570s
user    0m13.163s
sys     0m14.496s

```

Gambar 4.13: nixos-rebuild second target tanpa cache

Tabel 4.1: nixos-rebuild first target tanpa cache

Iterasi ke-	real	user	sys
1	3m29.511s	0m12.695s	0m14.709s
2	3m27.362s	0m12.970s	0m14.277s
3	3m28.368s	0m12.887s	0m14.580s

Tabel 4.2: nixos-rebuild second target tanpa cache

Iterasi ke-	real	user	sys
1	3m28.804s	0m12.822s	0m14.634s
2	3m27.731s	0m12.900s	0m14.437s
3	3m36.570s	0m13.163s	0m14.496s

1.b. NixOS dengan Cache

Pada percobaan ini, nixos-rebuild tidak lagi perlu mengunduh binary karena sudah di *build* di *master*. *Master* hanya perlu menyalin *closure* ke target. Percobaan dilakukan sebanyak tiga kali, dimana sebelum percobaan, VM dan target akan di *snapshot* sehingga dapat kembali ke *state* sebelum diterapkannya konfigurasi.

```

reloading the following units: dbus.service, reload-systemd-vconsole-setup.service
restarting the following units: boot.mount, dev-hugepages.mount, sshd.service, sys
starting the following units: NetworkManager-wait-online.service, NetworkManager.se
service, network-local-commands.service, network-setup.service, nscd.service, reso
d-oomd.socket, systemd-sysctl.service, systemd-timesyncd.service, systemd-tmpfiles-
e-done.service
the following new units were started: NetworkManager-dispatcher.service, firewal
l.service, phpfpm.slice, phpfpm.target, systemd-fsck@dev-disk-by\x2dlabel-BOOT.servic
eup.service

real    1m4.359s
user    0m3.129s
sys     0m6.261s

reloading the following units: dbus.service, reload-systemd-vconsole-setup.service
restarting the following units: boot.mount, dev-hugepages.mount, sshd.service, sys
starting the following units: NetworkManager-wait-online.service, NetworkManager.se
service, network-local-commands.service, network-setup.service, nscd.service, reso
d-oomd.socket, systemd-sysctl.service, systemd-timesyncd.service, systemd-tmpfiles-
e-done.service
the following new units were started: NetworkManager-dispatcher.service, firewal
l.service, phpfpm.slice, phpfpm.target, systemd-fsck@dev-disk-by\x2dlabel-BOOT.servic
eup.service

real    1m9.879s
user    0m3.297s
sys     0m7.239s

reloading the following units: dbus.service, reload-systemd-vconsole-setup.service
restarting the following units: boot.mount, dev-hugepages.mount, sshd.service, sys
starting the following units: NetworkManager-wait-online.service, NetworkManager.se
service, network-local-commands.service, network-setup.service, nscd.service, reso
d-oomd.socket, systemd-sysctl.service, systemd-timesyncd.service, systemd-tmpfiles-
e-done.service
the following new units were started: NetworkManager-dispatcher.service, firewal
l.service, phpfpm.slice, phpfpm.target, systemd-fsck@dev-disk-by\x2dlabel-BOOT.servic
eup.service

real    1m13.626s
user    0m3.179s
sys     0m6.938s

```

Gambar 4.14: *nixos-rebuild first target dengan cache*

```

reloading the following units: dbus.service, reload-syste
m-vconsole-setup.service
restarting the following units: boot.mount, dev-hugepages.mount, nix-daemon.service
starting the following units: NetworkManager-wait-online.service, NetworkManager.s
ervice, network-local-commands.service, network-setup.service, nscd.service, reso
lvd-oomd.socket, systemd-syctl.service, systemd-timesyncd.service, systemd-tmpfiles
d-done.service
the following new units were started: NetworkManager-dispatcher.service, firewal
l, phpfpm.target, systemd-fsck@dev-disk-by\x2dlabel-BOOT.service, systemd-hostnamed
real    1m12.569s
user    0m3.389s
sys    0m6.435s
reloading the following units: dbus.service, reload-syste
m-vconsole-setup.service
restarting the following units: boot.mount, dev-hugepages.mount, nix-daemon.service
starting the following units: NetworkManager-wait-online.service, NetworkManager.s
ervice, network-local-commands.service, network-setup.service, nscd.service, reso
lvd-oomd.socket, systemd-syctl.service, systemd-timesyncd.service, systemd-tmpfiles
d-done.service
the following new units were started: NetworkManager-dispatcher.service, firewal
l, phpfpm.target, systemd-fsck@dev-disk-by\x2dlabel-BOOT.service, systemd-hostnamed
real    1m12.321s
user    0m3.414s
sys    0m6.465s
reloading the following units: dbus.service, reload-syste
m-vconsole-setup.service
restarting the following units: boot.mount, dev-hugepages.mount, nix-daemon.service
starting the following units: NetworkManager-wait-online.service, NetworkManager.s
ervice, network-local-commands.service, network-setup.service, nscd.service, reso
lvd-oomd.socket, systemd-syctl.service, systemd-timesyncd.service, systemd-tmpfiles
d-done.service
the following new units were started: NetworkManager-dispatcher.service, firewal
l, phpfpm.target, systemd-fsck@dev-disk-by\x2dlabel-BOOT.service, systemd-hostnamed
real    1m13.510s
user    0m3.168s
sys    0m6.520s

```

Gambar 4.15: *nixos-rebuild second target dengan cache*Tabel 4.3: *nixos-rebuild first target dengan cache*

Iterasi ke-	real	user	sys
1	1m4.359s	0m3.129s	0m6.261s
2	1m9.078s	0m3.287s	0m7.239s
3	1m13.626s	0m3.178s	0m6.930s

Tabel 4.4: *nixos-rebuild second target*

Iterasi ke-	real	user	sys
1	1m12.569s	0m3.389s	0m6.435s
2	1m12.321s	0m3.414s	0m6.465s
3	1m13.510s	0m3.168s	0m6.520s

Patut diketahui bahwa kecepatan untuk menyalin *closure* dari *master* ke target bergantung kepada koneksi

antara keduanya. Dalam percobaan ini, antara *master* dengan target terkoneksi dengan *bandwidth* sebesar 1 Gbps.

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 dengan setiap kali dijalankan, VM target akan dikembalikan ke *snapshot* yang berisi *state* sebelum ansible-playbook diterapkan ke target. Percobaan ini dilakukan tiga kali di masing-masing target.

```

TASK [Gathering Facts] *****
ok: [192.168.100.27]

TASK [restart ssh] *****
changed: [192.168.100.27]

PLAY RECAP *****
192.168.100.27 : ok=19  changed=15  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

real   2m5.295s
user   0m7.874s
sys    0m2.862s
- 

TASK [restart ssh] *****
changed: [192.168.100.27]

PLAY RECAP *****
192.168.100.27 : ok=19  changed=15  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

real   1m54.364s
user   0m6.795s
sys    0m1.978s
- 

TASK [restart ssh] *****
changed: [192.168.100.27]

PLAY RECAP *****
192.168.100.27 : ok=19  changed=15  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

real   2m35.291s
user   0m7.690s
sys    0m2.189s
-
```

Gambar 4.16: Ansible first target pasca install

```

TASK [restart ssh] *****
changed: [(192.168.100.28)

PLAY RECAP *****
192.168.100.28 : ok=19 changed=15 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

real 2m27.849s
user 0m7.529s
sys 0m2.185s
TASK [restart ssh] *****
changed: [(192.168.100.29)

PLAY RECAP *****
192.168.100.29 : ok=19 changed=15 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

real 1m56.198s
user 0m5.801s
sys 0m2.185s
TASK [restart ssh] *****
changed: [(192.168.100.29)

PLAY RECAP *****
192.168.100.28 : ok=19 changed=15 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

real 2m4.840s
user 0m6.991s
sys 0m2.082s

```

Gambar 4.17: Ansible second target pasca install

Tabel 4.5: Ansible first target pasca install

Iterasi ke-	real	user	sys
1	2m5.295s	0m7.074s	0m2.062s
2	1m54.364s	0m6.795s	0m1.970s
3	2m35.291s	0m7.698s	0m2.109s

Tabel 4.6: ansible second target pasca install

Iterasi ke-	real	user	sys
1	2m32.026s	0m10.997s	0m3.703s
2	3m19.488s	0m12.124s	0m3.995s
3	4m23.604s	0m13.486s	0m4.481s

1.d. Ansible Multi Target

Pada percobaan ini, dijalankan menggunakan perintah ansible-playbook dengan dua target secara paralel. Setiap kali sebelum ansible-playbook diterapkan ke target, target akan dikembalikan ke *snapshot* sebelum ansible-playbook diterapkan.

```

PLAY RECAP ****
192.168.100.27 : ok=19  changed=15  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
192.168.100.28 : ok=19  changed=15  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

real   2m32.026s
user   0m10.997s
sys    0m3.703s
TASK [restart ssh] ****
changed: [192.168.100.27]
changed: [192.168.100.28]

PLAY RECAP ****
192.168.100.27 : ok=19  changed=15  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
192.168.100.28 : ok=19  changed=15  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

real   3m19.408s
user   0m12.124s
sys    0m3.995s
TASK [restart ssh] ****
changed: [192.168.100.27]
changed: [192.168.100.28]

PLAY RECAP ****
192.168.100.27 : ok=19  changed=15  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
192.168.100.28 : ok=19  changed=15  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

real   4m23.504s
user   0m13.496s
sys    0m4.491s

```

Gambar 4.18: Ansible multi target pasca install

Tabel 4.7: ansible multi target pasca install

Iterasi ke-	real	user	sys
1	2m12.327s	0m9.902s	0m3.188s
2	1m56.784s	0m9.654s	0m3.147s
3	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

Untuk NixOS, dijalankan perintah nixos-rebuild dengan dua target. Masing-masing target dijalankan sebanyak tiga kali tanpa perlu mengembalikan VM ke

kondisi sebelumnya dengan *snapshot*. Berikut hasil pengujianya:

```
copying 0 paths...
activating the configuration...
setting up /etc...
reloading user units for root...
reloading user units for rizqirazkafi...
setting up tmpfiles

real    0m6.881s
user    0m0.241s
sys     0m0.088s
copying 0 paths...
activating the configuration...
setting up /etc...
reloading user units for root...
reloading user units for rizqirazkafi...
setting up tmpfiles

real    0m3.998s
user    0m0.223s
sys     0m0.088s
copying 0 paths...
activating the configuration...
setting up /etc...
reloading user units for root...
reloading user units for rizqirazkafi...
setting up tmpfiles

real    0m2.899s
user    0m0.227s
sys     0m0.086s
```

Gambar 4.19: nixos-rebuild first target pasca install rerun

```
copying 0 paths...
activating the configuration...
setting up /etc...
reloading user units for root...
reloading user units for rizqirazkafi...
setting up tmpfiles

real    0m7.471s
user    0m0.247s
sys     0m0.089s

copying 0 paths...
activating the configuration...
setting up /etc...
reloading user units for root...
reloading user units for rizqirazkafi...
setting up tmpfiles

real    0m4.524s
user    0m0.231s
sys     0m0.078s

copying 0 paths...
activating the configuration...
setting up /etc...
reloading user units for root...
reloading user units for rizqirazkafi...
setting up tmpfiles

real    0m2.802s
user    0m0.235s
sys     0m0.079s
```

Gambar 4.20: nixos-rebuild second target pasca install rerun

Tabel 4.8: nixos-rebuild first target pasca install rerun

Iterasi ke-	real	user	sys
1	0m6.881s	0m0.241s	0m0.088s
2	0m3.998s	0m0.223s	0m0.088s
3	0m2.899s	0m0.227s	0m0.086s

Tabel 4.9: nixos-rebuild second target pasca install rerun

Iterasi ke-	real	user	sys
1	0m7.471s	0m0.247s	0m0.089s
2	0m4.524s	0m0.231s	0m0.078s
3	0m2.802s	0m0.235s	0m0.079s

2.b. Ansible single target

Untuk percobaan Ansible *single target*, perintah ansible-playbook dijalankan ke satu target dalam satu waktu. Ini digunakan untuk mendapatkan hasil perbandingan yang setara dengan nixos-rebuild yang hanya mampu mengkonfigurasi satu target dalam satu waktu.

```

TASK [restart ssh] *****
changed: (192.168.100.27)

PLAY RECAP *****
192.168.100.27 : ok=19 changed=3 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

real 0m16.825s
user 0m4.369s
sys 0m1.168s
TASK [restart ssh] *****
changed: (192.168.100.27)

PLAY RECAP *****
192.168.100.27 : ok=19 changed=3 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

real 0m15.725s
user 0m4.278s
sys 0m1.122s
TASK [restart ssh] *****
changed: (192.168.100.27)

PLAY RECAP *****
192.168.100.27 : ok=19 changed=3 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

real 0m15.593s
user 0m4.278s
sys 0m1.095s

```

Gambar 4.21: Ansible first target rerun

```

TRGK [restart ssh] *****
changed: [(192.168.100.20)]

PLAY RECAP *****
192.168.100.20 : ok=19  changed=3  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

real   0m15.709s
user   0m4.323s
sys    0m1.083s
TRGK [restart ssh] *****
changed: [(192.168.100.20)]

PLAY RECAP *****
192.168.100.20 : ok=19  changed=3  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

real   0m15.445s
user   0m4.272s
sys    0m1.089s
TRGK [restart ssh] *****
changed: [(192.168.100.20)]

PLAY RECAP *****
192.168.100.20 : ok=19  changed=3  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

real   0m15.491s
user   0m4.253s
sys    0m1.086s

```

Gambar 4.22: Ansible second target rerun

Tabel 4.10: Ansible first target rerun

Iterasi ke-	real	user	sys
1	0m16.025s	0m4.309s	0m1.108s
2	0m15.725s	0m4.270s	0m1.122s
3	0m15.583s	0m4.270s	0m1.096s

Tabel 4.11: Ansible second target rerun

Iterasi ke-	real	user	sys
1	0m15.709s	0m4.323s	0m1.083s
2	0m15.445s	0m4.272s	0m1.089s
3	0m15.491s	0m4.253s	0m1.086s

2.c. Ansible multi target

Untuk percobaan Ansible *multi target*, perintah `ansible-playbook` dijalankan ke dua target secara paralel. Berikut merupakan hasil dari percobaan Ansible *multi target*:

```

TRK [restart ssh] *****
changed: [192.168.100.20]
changed: [192.168.100.27]

PLAY RECAP *****
192.168.100.27 : ok=19 changed=3 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
192.168.100.20 : ok=19 changed=3 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

real 0m17.521s
user 0m6.973s
sys 0m2.093s
TRK [restart ssh] *****
changed: [192.168.100.20]
changed: [192.168.100.27]

PLAY RECAP *****
192.168.100.27 : ok=19 changed=3 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
192.168.100.20 : ok=19 changed=3 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

real 0m17.259s
user 0m6.898s
sys 0m2.197s
TRK [restart ssh] *****
changed: [192.168.100.27]
changed: [192.168.100.20]

PLAY RECAP *****
192.168.100.27 : ok=19 changed=3 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
192.168.100.20 : ok=19 changed=3 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

real 0m17.250s
user 0m6.837s
sys 0m2.196s

```

Gambar 4.23: Ansible multi target rerun

Tabel 4.12: Ansible multi target

Iterasi ke-	real	user	sys
1	0m17.521s	0m6.973s	0m2.093s
2	0m17.259s	0m6.898s	0m2.137s
3	0m17.250s	0m6.837s	0m2.196s

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

Untuk NixOS, penambahan konfigurasi dilakukan pada file configuration.nix, pada bagian environment.systemPackages sebagai berikut:

```

environment.systemPackages = with pkgs; [
    go
];

```

Listing 4.11: NixOS Go Package

Pengujian dilakukan dengan pertama mengembalikan kondisi sistem baik *master* maupun target ke konfigurasi setelah pasca install, kemudian menambahkan "go" kedalam konfigurasi. Ini dimaksudkan agar NixOS tidak menyimpan cache hasil *build* untuk paket Go. Konfigurasi kemudian diterapkan dengan nixos-rebuild oleh *master* ke target. Setelah *master* melakukan *build*, *master* kemudian menyalin hasilnya ke target. Percobaan ini dilakukan berulang sebanyak tiga kali per target.

Gambar 4.24: nixos-rebuild first target install Go

Gambar 4.25: *nixos-rebuild second target install Go*

Tabel 4.13: *nixos-rebuild* first target

Iterasi ke-	real	user	sys
1	0m39.727s	0m7.400s	0m2.589s
2	0m38.222s	0m7.297s	0m2.637s
3	0m40.365s	0m7.118s	0m2.905s

Tabel 4.14: *nixos-rebuild second target*

Iterasi ke-	real	user	sys
1	0m39.300s	0m7.285s	0m2.546s
2	0m39.190s	0m7.257s	0m2.687s
3	0m39.117s	0m7.305s	0m2.556s

3.b. NixOS dengan cache

Pada percobaan ini, NixOS telah melakukan *build* untuk paket GO dan nixos-rebuild hanya perlu menyalin hasilnya ke target. Dari percobaan tersebut, didapatkan hasil sebagai berikut dengan pertimbangan *bandwidth* dari *master* ke target adalah 1 Gbps.

Gambar 4.26: nixos-rebuild first target install Go cache

```

copying 12 paths...
copying path '/nix/store/we97rhhdzacb158y0crpvg517zqbnsl-l-go-1.21.5' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/rphmr3rljg1i806nluxdm95drdgdkbv-system-path' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/b931vh9f1xwar3axug3gl08czugnx-j-dbus-1' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/qdk93jcf19r4gb43jrvarjr40nzyus-X-Restart-Triggers-polkit' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/zf1k922k1zslr27y3qgcb0j8c295vuw-X-Restart-Triggers-dbus' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/d8567mew2k5jczs2q40w2vz3nlvls9s-unit-polkit.service' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/advn26vdxr3r81vql16p26can27hmp9-unit-dbus.service' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/97bk5k7g-9r6sygvz2l99kxp7480l1jyk-vunit-dbus.service' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/5e3a1d1s2zghp938vnb22jhdkv415-user-units' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/8kdn75knk9a7c54ebelbgmab9s0bzv-system-units' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/76u40gpx2240h291jpb67652c8kjx-n-etc' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/n61d917b89jls8kq3wdzddak7czggz-nixos-system-nixos-vm-23.11.28240223.c5101e4' to 'ssh://root@192.168.100.26'...
activating the configuration...
setting up /etc...
reloading user units for root...
reloading user units for rizqirazkafi...
setting up tmfiles
reloading the following units: dbus.service

real    0m11.527s
user    0m0.404s
sys     0m0.470s

copying 12 paths...
copying path '/nix/store/we97rhhdzacb158y0crpvg517zqbnsl-l-go-1.21.5' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/rphmr3rljg1i806nluxdm95drdgdkbv-system-path' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/b931vh9f1xwar3axug3gl08czugnx-j-dbus-1' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/qdk93jcf19r4gb43jrvarjr40nzyus-X-Restart-Triggers-polkit' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/zf1k922k1zslr27y3qgcb0j8c295vuw-X-Restart-Triggers-dbus' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/d8567mew2k5jczs2q40w2vz3nlvls9s-unit-polkit.service' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/97bk5k7g-9r6sygvz2l99kxp7480l1jyk-vunit-dbus.service' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/advn26vdxr3r81vql16p26can27hmp9-unit-dbus.service' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/5e3a1d1s2zghp938vnb22jhdkv415-user-units' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/8kdn75knk9a7c54ebelbgmab9s0bzv-system-units' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/76u40gpx2240h291jpb67652c8kjx-n-etc' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/n61d917b89jls8kq3wdzddak7czggz-nixos-system-nixos-vm-23.11.28240223.c5101e4' to 'ssh://root@192.168.100.26'...
activating the configuration...
setting up /etc...
reloading user units for root...
reloading user units for rizqirazkafi...
setting up tmfiles
reloading the following units: dbus.service

real    0m14.384s
user    0m0.492s
sys     0m0.454s

copying 12 paths...
copying path '/nix/store/we97rhhdzacb158y0crpvg517zqbnsl-l-go-1.21.5' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/rphmr3rljg1i806nluxdm95drdgdkbv-system-path' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/b931vh9f1xwar3axug3gl08czugnx-j-dbus-1' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/qdk93jcf19r4gb43jrvarjr40nzyus-X-Restart-Triggers-polkit' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/zf1k922k1zslr27y3qgcb0j8c295vuw-X-Restart-Triggers-dbus' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/d8567mew2k5jczs2q40w2vz3nlvls9s-unit-polkit.service' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/97bk5k7g-9r6sygvz2l99kxp7480l1jyk-vunit-dbus.service' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/advn26vdxr3r81vql16p26can27hmp9-unit-dbus.service' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/5e3a1d1s2zghp938vnb22jhdkv415-user-units' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/8kdn75knk9a7c54ebelbgmab9s0bzv-system-units' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/76u40gpx2240h291jpb67652c8kjx-n-etc' to 'ssh://root@192.168.100.26'...
copying path '/nix/store/n61d917b89jls8kq3wdzddak7czggz-nixos-system-nixos-vm-23.11.28240223.c5101e4' to 'ssh://root@192.168.100.26'...
setting up /etc...
activating the configuration...
setting up /etc...
reloading user units for root...
reloading user units for rizqirazkafi...
setting up tmfiles
reloading the following units: dbus.service

real    0m13.689s
user    0m0.491s
sys     0m0.475s

```

Gambar 4.27: nixos-rebuild second target install Go cache

Tabel 4.15: nixos-rebuild first target install Go cache

Iterasi ke-	real	user	sys
1	0m14.169s	0m0.476s	0m0.523s
2	0m13.795s	0m0.505s	0m0.451s
3	0m14.796s	0m0.485s	0m0.464s

Tabel 4.16: nixos-rebuild second target install Go cache

Iterasi ke-	real	user	sys
1	0m11.527s	0m0.484s	0m0.470s
2	0m14.384s	0m0.492s	0m0.454s
3	0m13.683s	0m0.481s	0m0.475s

3.c. Ansible single target

Pada percobaan Ansible *single target*, penambahan paket "GO" di definisikan dalam file yml yang digunakan seperti berikut:

```
- name: install go
  ansible.builtin.apt:
    pkg:
      - golang-go
    state: present
```

Listing 4.12: Ansible penambahan paket GO

Pada konfigurasi tersebut, "state: absent" digunakan untuk memberitahu Ansible untuk memastikan apakah paket "golang-go" benar-benar ada pada sistem. Percobaan ini dilakukan sebanyak tiga kali untuk masing-masing target dan setiap sebelum percobaan kondisi VM akan dikembalikan ke *snapshot* pasca install.

```

[TRGK] [restart ssh] *****
changed: [(192.168.100.27)]

PLAY RECAP *****
192.168.100.27 : ok=19  changed=4  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

real   0m3d.498s
user   0m4.762s
sys    0m1.237s
[TRGK] [restart ssh] *****
changed: [(192.168.100.27)]

PLAY RECAP *****
192.168.100.27 : ok=19  changed=4  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

real   0m26.016s
user   0m4.552s
sys    0m1.165s
[TRGK] [restart ssh] *****
changed: [(192.168.100.27)]

PLAY RECAP *****
192.168.100.27 : ok=19  changed=4  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

real   0m26.752s
user   0m4.584s
sys    0m1.180s

```

Gambar 4.28: Ansible first target install Go

```

[TRGK] [restart ssh] *****
changed: [(192.168.100.29)]

PLAY RECAP *****
192.168.100.29 : ok=19  changed=4  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

real   0m26.308s
user   0m4.540s
sys    0m1.149s
[TRGK] [restart ssh] *****
changed: [(192.168.100.29)]

PLAY RECAP *****
192.168.100.29 : ok=19  changed=4  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

real   0m26.564s
user   0m4.534s
sys    0m1.219s
[TRGK] [restart ssh] *****
changed: [(192.168.100.29)]

PLAY RECAP *****
192.168.100.29 : ok=19  changed=4  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

real   0m25.938s
user   0m4.564s
sys    0m1.161s

```

Gambar 4.29: Ansible second target install Go

Tabel 4.17: ansible first target

Iterasi ke-	real	user	sys
1	0m3d.498s	0m4.762s	0m1.237s
2	0m26.016s	0m4.552s	0m1.165s
3	0m26.752s	0m4.584s	0m1.180s

Tabel 4.18: ansible second target

Iterasi ke-	real	user	sys
1	0m26.637s	0m4.281s	0m1.120s
2	0m26.256s	0m4.327s	0m1.027s
3	0m25.256s	0m4.310s	0m1.046s

3.d. Ansible multi target

Pada percobaan Ansible *multi target*, sama seperti percobaan Ansible *single target* pada bagian konfigurasi dan *snapshot*. Namun yang membedakan adalah, perintah ansible-playbook dijalankan pada dua target secara paralel. Berikut untuk hasilnya:

```

TASK [restart ssh] *****
changed: [(192.168.100.28)]
changed: [(192.168.100.27)]

PLAY RECAP *****
192.168.100.27 : ok=19 changed=4 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
192.168.100.28 : ok=19 changed=4 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

real   0m29.149s
user   0m7.147s
sys    0m2.236s
TASK [restart ssh] *****
changed: [(192.168.100.28)]
changed: [(192.168.100.27)]

PLAY RECAP *****
192.168.100.27 : ok=19 changed=4 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
192.168.100.28 : ok=19 changed=4 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

real   0m33.852s
user   0m7.427s
sys    0m2.386s
TASK [restart ssh] *****
changed: [(192.168.100.28)]
changed: [(192.168.100.27)]

PLAY RECAP *****
192.168.100.27 : ok=19 changed=4 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
192.168.100.28 : ok=19 changed=4 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

real   0m33.841s
user   0m7.393s
sys    0m2.407s

```

Gambar 4.30: Ansible multi target install Go

Tabel 4.19: Ansible multi target install Go

Iterasi ke-	real	user	sys
1	0m29.149s	0m7.147s	0m2.230s
2	0m33.052s	0m7.427s	0m2.308s
3	0m33.041s	0m7.393s	0m2.407s

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

Pada pengujian ini, deskripsi paket Go akan dihapus atau dikomen pada file konfigurasi configuration.nix seperti berikut:

```
environment.systemPackages = with pkgs; [
  # go
];
```

Listing 4.13: Penghapusan paket GO pada konfigurasi NixOS

Perintah nixos-rebuild kemudian dijalankan pada masing-masing target sebanyak tiga kali dan hasilnya sebagai berikut:

```
copying 0 paths...
activating the configuration...
setting up /etc...
reloading user units for root...
reloading user units for rizqirazkafi...
setting up tmpfiles
reloading the following units: dbus.service

real    0m3.299s
user    0m0.236s
sys     0m0.072s

copying 0 paths...
activating the configuration...
setting up /etc...
reloading user units for root...
reloading user units for rizqirazkafi...
setting up tmpfiles

real    0m3.088s
user    0m0.231s
sys     0m0.080s

copying 0 paths...
activating the configuration...
setting up /etc...
reloading user units for root...
reloading user units for rizqirazkafi...
setting up tmpfiles

real    0m2.886s
user    0m0.224s
sys     0m0.092s
```

Gambar 4.31: nixos-rebuild first target uninstall Go

```
copying 0 paths...
activating the configuration...
setting up /etc...
reloading user units for root...
reloading user units for rizqirazkafi...
setting up tmpfiles
reloading the following units: dbus.service

real    0m3.046s
user    0m0.241s
sys     0m0.068s

copying 0 paths...
activating the configuration...
setting up /etc...
reloading user units for root...
reloading user units for rizqirazkafi...
setting up tmpfiles

real    0m2.913s
user    0m0.233s
sys     0m0.082s

copying 0 paths...
activating the configuration...
setting up /etc...
reloading user units for root...
reloading user units for rizqirazkafi...
setting up tmpfiles

real    0m2.819s
user    0m0.229s
sys     0m0.082s
```

Gambar 4.32: nixos-rebuild second target uninstall Go

Tabel 4.20: nixos-rebuild first target uninstall Go

Iterasi ke-	real	user	sys
1	0m3.299s	0m0.236s	0m0.072s
2	0m3.088s	0m0.231s	0m0.080s
3	0m2.886s	0m0.224s	0m0.092s

Tabel 4.21: nixos-rebuild second target uninstall Go

Iterasi ke-	real	user	sys
1	0m3.046s	0m0.241s	0m0.068s
2	0m2.913s	0m0.233s	0m0.082s
3	0m2.819s	0m0.229s	0m0.082s

4.b. Ansible single target

Untuk Ansible, kita hanya perlu mengganti "state" untuk paket Go dari "present" menjadi "absent". Dengan ini kita memberitahu Ansible untuk memastikan paket "golang-go" tidak ada pada target. Perintah ansible-playbook kemudian dijalankan pada satu target dalam satu waktu sebanyak tiga kali dengan hasil dan konfigurasinya adalah sebagai berikut:

```
- name: install go
  ansible.builtin.apt:
    pkg:
      - golang-go
    state: absent
```

Listing 4.14: Konfigurasi penghapusan paket Go pada Ansible

```

TASK [restart ssh] *****
changed: [192.168.100.27]

PLAY RECAP *****
192.168.100.27 : ok=19  changed=4    unreachable=0    failed=0   skipped=0   rescued=0   ignored=0

real  0m18.857s
user  0m4.495s
sys   0m1.113s

TASK [restart ssh] *****
changed: [192.168.100.27]

PLAY RECAP *****
192.168.100.27 : ok=19  changed=4    unreachable=0    failed=0   skipped=0   rescued=0   ignored=0

real  0m19.608s
user  0m4.495s
sys   0m1.145s

TASK [restart ssh] *****
changed: [192.168.100.27]

PLAY RECAP *****
192.168.100.27 : ok=19  changed=4    unreachable=0    failed=0   skipped=0   rescued=0   ignored=0

real  0m19.679s
user  0m4.413s
sys   0m1.162s

```

Gambar 4.33: Ansible first target uninstall Go

```

TASK [restart ssh] *****
changed: [192.168.100.28]

PLAY RECAP *****
192.168.100.28 : ok=19  changed=4    unreachable=0    failed=0   skipped=0   rescued=0   ignored=0

real  0m19.791s
user  0m4.396s
sys   0m1.113s

PLAY RECAP *****
192.168.100.28 : ok=19  changed=4    unreachable=0    failed=0   skipped=0   rescued=0   ignored=0

real  0m18.061s
user  0m4.349s
sys   0m1.139s

TASK [restart ssh] *****
changed: [192.168.100.28]

PLAY RECAP *****
192.168.100.28 : ok=19  changed=4    unreachable=0    failed=0   skipped=0   rescued=0   ignored=0

real  0m18.089s
user  0m4.417s
sys   0m1.090s

```

Gambar 4.34: Ansible second target uninstall Go

Tabel 4.22: ansible first target

Iterasi ke-	real	user	sys
1	0m18.857s	0m4.435s	0m1.113s
2	0m19.608s	0m4.435s	0m1.145s
3	0m18.670s	0m4.413s	0m1.162s

Tabel 4.23: ansible second target

Iterasi ke-	real	user	sys
1	0m18.791s	0m4.336s	0m1.113s
2	0m18.861s	0m4.348s	0m1.133s
3	0m18.893s	0m4.417s	0m1.090s

4.c. Ansible multi target

Untuk percobaan Ansible *multi target* sama seperti Ansible *single target*. Pembedanya terletak hanya di jumlah target yang dituju dimana untuk Ansible *multi target* adalah sebanyak dua target dalam percobaan ini. Berikut adalah hasil dari percobaan ini:

```

[TRGK [restart ssh] *****
changed: [192.168.100.26]
changed: [192.168.100.27]

PLAY RECAP *****
192.168.100.27 : ok=19  changed=4    unreachable=0    failed=0  skipped=0   rescued=0   ignored=0
192.168.100.28 : ok=19  changed=4    unreachable=0    failed=0  skipped=0   rescued=0   ignored=0

real  0m20.943s
user  0m6.971s
sys   0m2.246s
[TRGK [restart ssh] *****
changed: [192.168.100.27]
changed: [192.168.100.28]

PLAY RECAP *****
192.168.100.27 : ok=19  changed=4    unreachable=0    failed=0  skipped=0   rescued=0   ignored=0
192.168.100.28 : ok=19  changed=4    unreachable=0    failed=0  skipped=0   rescued=0   ignored=0

real  0m22.876s
user  0m7.190s
sys   0m2.225s
[TRGK [restart ssh] *****
changed: [192.168.100.27]
changed: [192.168.100.28]

PLAY RECAP *****
192.168.100.27 : ok=19  changed=4    unreachable=0    failed=0  skipped=0   rescued=0   ignored=0
192.168.100.28 : ok=19  changed=4    unreachable=0    failed=0  skipped=0   rescued=0   ignored=0

real  0m21.062s
user  0m7.023s
sys   0m2.289s

```

Gambar 4.35: Ansible multi target uninstall Go

Tabel 4.24: ansible multi target

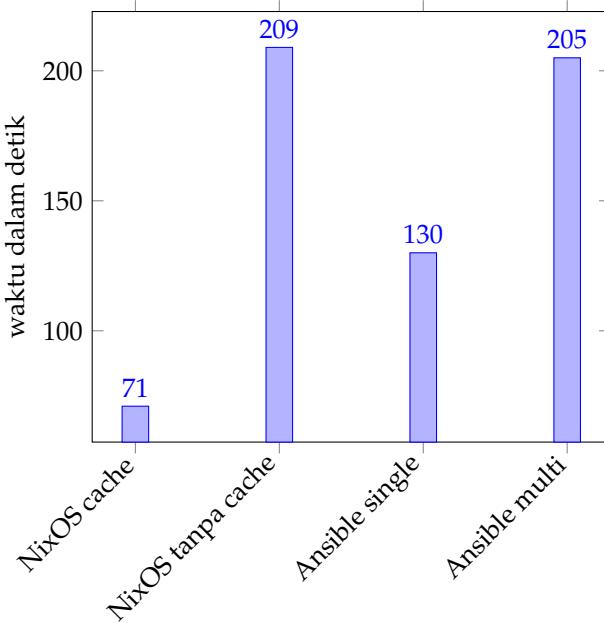
Iterasi ke-	real	user	sys
1	0m20.943s	0m6.971s	0m2.246s
2	0m22.876s	0m7.190s	0m2.225s
3	0m21.062s	0m7.023s	0m2.289s

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.

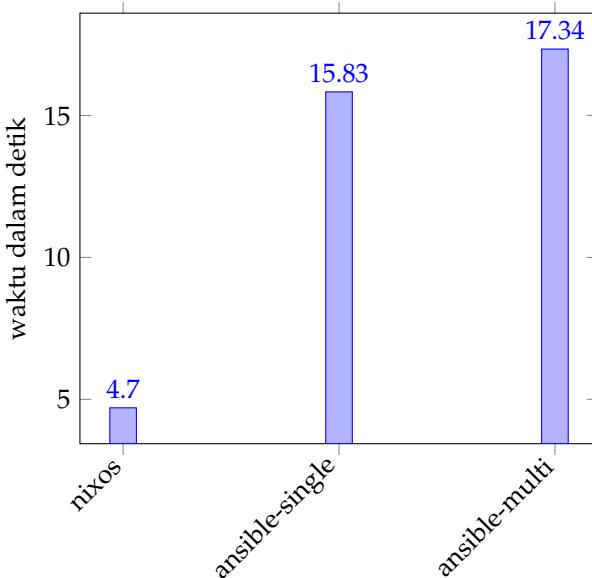


Dapat kita lihat bahwa NixOS tanpa cache butuh waktu lebih lama dikarenakan sistem *package manager* yang akan mengunduh *dependency* tiap paket secara terpisah apabila ada perbedaan versi. Faktor lain adalah tidak adanya sistem mirror seperti yang dimiliki oleh repository Ubuntu dimana paket bisa di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

Untuk Ansible terutama untuk multi target memang rata-rata waktu lebih lama. Namun melihat hasil yang didapat dari tiga kali percobaan, waktunya sangat fluktuatif dan tidak konsisten. Namun apabila kita tambahkan hasil dari NixOS tanpa cache dengan NixOS dengan cache, maka total dari rata-ratanya adalah 280 detik untuk sekali *build* dan dua kali *copy*. Waktu eksekusi untuk NixOS tentu akan bertambah

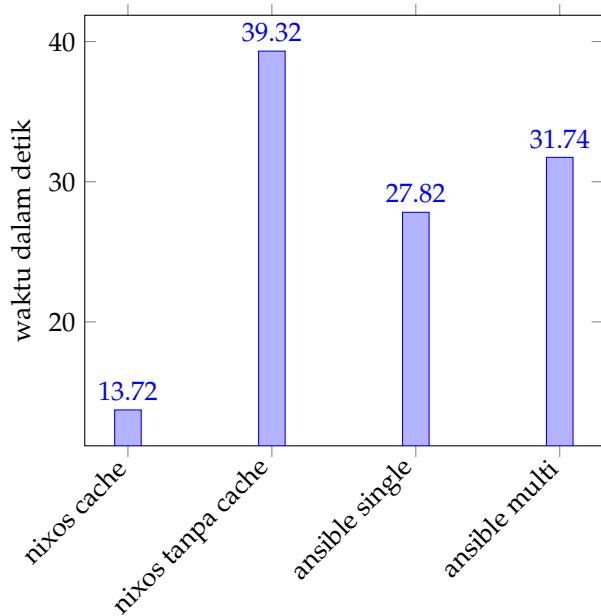
seiring bertambahnya target yang perlu di konfigurasi pada satu waktu. Berbeda dengan Ansible yang dapat bekerja secara paralel.

5.b. Penerapan ulang setelah pasca install



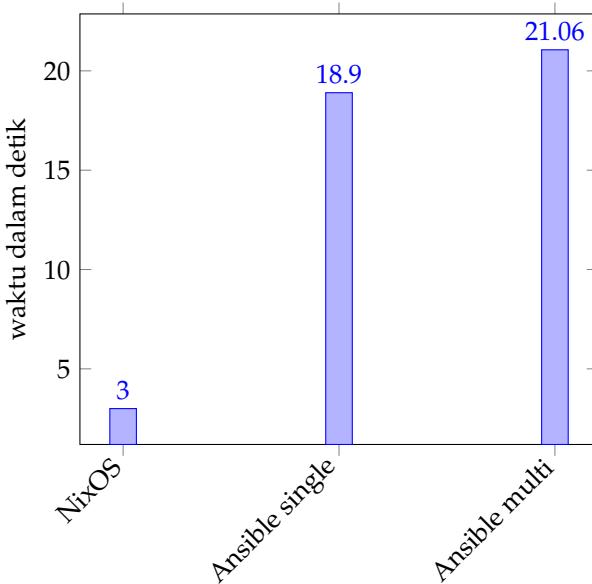
Disini dapat dilihat Ansible membutuhkan waktu lebih lama untuk melakukan pengecekan dikarenakan Ansible melakukan pengecekan tiap *task* pada setiap target sedangkan NixOS hanya mengecek perubahan pada konfigurasi dan membandingkan *closure* yang ada pada target dengan *host*.

5.c. Penambahan paket GO



Pada grafik diatas, dapat dilihat lagi-lagi NixOS dengan cache lebih unggul untuk kasus penambahan paket GO untuk satu target.

5.d. Penghapusan paket GO



Pada grafik diatas dapat terlihat bahwa untuk menghapus paket, NixOS lebih cepat daripada Ansible. Ini disebabkan karena NixOS tidak langsung menghapus paket dari sistem, melainkan hanya menghapus *symlink*. Binary tetap disimpan apabila di kemudian hari paket dibutuhkan untuk diinstall kembali. NixOS hanya menghapus paket apabila *garbage collector* di eksekusi.

6. Hasil Konfigurasi

6.a. NixOS

Dapat dilihat bahwa NixOS berhasil mengkonfigurasi target dengan paket dan konfigurasi *service* seperti Nginx dan SSH.

Gambar 4.36: Hasil konfigurasi NixOS

6.b. Ansible

Hasil dari Ansible juga menunjukkan bahwa Ansible mampu mengkonfigurasi Ubuntu Server sesuai dengan konfigurasi dari keinginan penulis seperti dibawah ini

Gambar 4.37: Hasil konfigurasi Ansible

7. Deklaratif dan Imperatif

Dalam percobaan diatas, terdapat beberapa hal yang ditemui oleh pengujii tentang *tool* manajemen

konfigurasi. Ansible Playbook menggunakan metode imperatif dalam praktiknya, dimana Ansible Playbook berisi mengenai apa-apa saja yang harus dilakukan oleh Ansible. Ini menyebabkan ketidak cocokan antara Ansible Playbook dengan kondisi akhir sistem. Apabila kita mendeskripsikan *task* untuk menginstall dan menjalankan Nginx lalu kita terapkan, maka Nginx akan terinstall. Namun apabila *task* tersebut dihapus dari Ansible Playbook kemudian kita jalankan, maka Nginx akan tetap berjalan.

Pada sisi lain, konfigurasi NixOS merupakan representasi dari kondisi akhir sistem. Apabila kita pada awalnya mendeskripsikan *service* Nginx, maka Nginx akan berjalan pada sistem. Apabila kita menghapus pendeskripsi Nginx dari file konfigurasi, maka Nginx tidak akan berjalan. Metode inilah yang disebut dengan Deklaratif oleh NixOS.

Sebagai berikut contohnya:

```
{
imports = [
# Include the results of the hardware scan.
./hardware-configuration.nix
./vim.nix
inputs.home-manager.nixosModules.home-manager
./nginx.nix
];
}
```

Maka Nginx akan di konfigurasi dan berjalan di sistem target seperti berikut:

```
[root@rizarackafl0nixos-vm ~]# journalctl -f
* nginx.service - Nginx Web Server
  Loaded: loaded (/etc/systemd/system/nginx.service; enabled; preset: enabled)
  Active: active (running) since Thu 2024-05-23 21:29:10 WIB; 2 weeks 6 days ago
    Main PID: 2897 (nginx)
       CPU: 0.000 CPU: 0.000
      Tasks: 2 (limit: 4099)
     Memory: 5.4M
        CPU: 10ms
       CGroup: /system.slice/nginx.service
           ├─2897 nginx: master process /nix/store/lb91dad65044ek5920blyhdpgj7epa30-nginx-1.24.0/bin/nginx
           └─2898 nginx: worker process

May 23 21:29:18 nixos systemd[1]: Starting Nginx Web Server...
May 23 21:29:18 nixos nginx-pre-start[2543]: nginx: the configuration file /nix/store/j553karl4j7o9sx1zhdjbxb285xk2g
May 23 21:29:18 nixos nginx-pre-start[2543]: nginx: configuration file /nix/store/j553karl4j7o9sx1zhdjbxb285xk2g
May 23 21:29:18 nixos systemd[1]: Started Nginx Web Server.
lines 1-17/17 (END)
```

Gambar 4.38: Nix Nginx enable

Dan apabila kita ubah sebagai berikut:

```
{
imports = [
# Include the results of the hardware scan.
./hardware-configuration.nix
./vim.nix
inputs.home-manager.nixosModules.home-manager
# ./nginx.nix
];
}
```

Maka hasilnya akan seperti berikut dimana bahkan service Nginx dihapus dari sistem.

```
[rizqirazkafi@nixos-vm:~]$ systemctl status nginx
Unit nginx.service could not be found.
```

Gambar 4.39: Nix Nginx removed

Sedangkan untuk Ansible:

```
tasks:
- name: install packages
  ansible.builtin.apt:
    pkg:
      - nginx
tasks:
- name: enable nginx
  ansible.builtin.service:
    name: nginx
    state: restarted
```

Maka hasilnya akan sebagai berikut:

```
[rizqirazkafi@ansible-target-1:~]$ systemctl status nginx
nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Thu 2024-06-13 07:48:21 UTC; 5h 29min ago
     Docs: man:nginx(8)
Main PID: 27793 (nginx)
   Tasks: 3 (limit: 4686)
  Memory: 2.41 (peak: 2.8M)
    CPU: 0.000 CPU(s) used
   CGroup: /system.slice/nginx.service
           ├─27793 "nginx master process /usr/sbin/nginx -g daemon on; master_process on;"
           └─27794 "nginx worker process"
              ├─27795 "nginx worker process"

Jun 13 07:48:21 ansible-target-1 systemd[1]: Starting nginx.service - A high performance web server and a rever
Jun 13 07:48:21 ansible-target-1 systemd[1]: Started nginx.service - A high performance web server and a rever
```

Gambar 4.40: Enable nginx

Dan apabila baris diatas kita beri komentar didepannya, maka hasilnya sebagai berikut:

```
* nginx.service - A high performance web server and a reverse proxy server
  Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
  Active: active (running) since Thu 2024-06-13 07:48:21 UTC; 5h 31min ago
    Docs: man:nginx(8)
 Main PID: 27793 (nginx)
   Tasks: 3 (limit: 4006)
  Memory: 2.4M (peak: 2.6M)
    CPU: 20ms
   CGroup: /system.slice/nginx.service
           ├─27793 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
           ├─27794 "nginx: worker process"
           ├─27795 "nginx: worker process"
Jun 13 07:48:21 ansible-target-1 systemd[1]: Starting nginx.service - A high performance web server and a revers
Jun 13 07:48:21 ansible-target-1 systemd[1]: Started nginx.service - A high performance web server and a revers
```

Gambar 4.41: Nginx masih berjalan

Hanya jika kita mengganti state dari "enable nginx" ke "stopped" dan mendefinisikan "state" dari pkg Nginx ke "absent" barulah Nginx dihentikan dan dihapus dari sistem.

BAB V

KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan penelitian perbandingan performa waktu eksekusi otomasi 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 mendapatkan berbagai poin berikut:
 - a. Ansible mampu mengkonfigurasi sistem operasi berbasis GNU/Linux (pada kasus ini Ubuntu Server), namun hanya mampu pada tingkat imperatif dan bukan deklaratif.
 - b. NixOS dengan nixos-rebuild mampu mengkonfigurasi sistem operasi berbasis GNU/Linux (pada kasus ini NixOS) secara deklaratif.
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 *multi* 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 /closure* 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. *MATRIX : 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 jtik* [Doctoral dissertation]. Retrieved March 17, 2024, from <https://repository.pnj.ac.id/id/eprint/12406/1/Halaman%20Identitas%20Skripsi.pdf>