Published online on the page: https://journal.upp.ac.id/index.php/rjti/index

R J T I

Riau Jurnal Teknik Informatika

| E - ISSN (*Online*) 2829-2529 |

Implementasi Sistem Otomatisasi Konfigurasi *Server* Berbasis *Ansible* dan *Semaphore* pada Infrastruktur TI Perusahaan Ritel

Arie Fadhudin[1*], Desi Ramayanti[2]

[1,2] Teknik Informatika Fakultas Teknik dan Informatika Universitas Dian Nusantara, Jakarta Barat, Indonesia

Informasi Artikel

Sejarah Artikel:
Submit: 14 Desember 2025
Revisi: 06 Februari 2026
Diterima: 25 Februari 2026
Diterbitkan: 24 Maret 2026

Kata Kunci

Ansible, Semaphore, Infrastructure as Code, Server Automation, DNS, NTP

Korespondensi

E-mail: 411212015@mahasiswa.undira.ac.id*

A B S T R A C T

Manual configuration management of server infrastructure in retail companies with multiple branches often faces challenges related to time inefficiency and a high risk of human error. This study aims to implement an automated configuration system for DNS and NTP server s using the Infrastructure as Code (IaC) approach, utilizing Ansible as the automation tool and the Semaphore management interface. The research method employed is experimental with a Pre-test and Post-test Group design combined with performance testing and functional validation to compare the performance of manual (conventional) configuration and automated configuration. The results indicate a significant improvement in efficiency. In the single-server scenario, configuration time was reduced from 27 minutes to 8 minutes, achieving an efficiency improvement of 70.4%. A more substantial performance gain was observed in the large-scale implementation involving 14 server s, where the automation approach reduced the estimated configuration time from 378 minutes (manual method) to only 9 minutes through parallel execution, resulting in a time efficiency of 97.6%. In addition to improving speed, the proposed system standardizes configurations, eliminates technical errors, and facilitates centralized infrastructure monitoring.

Abstrak

Manajemen konfigurasi infrastruktur server secara manual pada perusahaan ritel dengan banyak cabang sering kali menghadapi kendala efisiensi waktu dan tingginya risiko kesalahan manusia (human error). Penelitian ini bertujuan untuk mengimplementasikan sistem otomatisasi konfigurasi server DNS dan NTP menggunakan pendekatan Infrastructure as Code (IaC) dengan tools Ansible dan antarmuka manajemen Semaphore. Metode penelitian yang digunakan adalah eksperimental dengan desain Pre-test and Post-test Group serta pengujian kinerja (performance testing) dan validasi fungsional untuk membandingkan kinerja antara konfigurasi manual (konvensional) dan otomatisasi. Hasil pengujian menunjukkan peningkatan efisiensi yang signifikan. Pada skenario satu server, waktu konfigurasi berkurang dari 27 menit menjadi 8 menit dengan efisiensi sebesar 70,4%. Peningkatan kinerja yang lebih drastis terlihat pada implementasi skala luas (14 server), di mana metode otomatisasi mampu memangkas waktu dari estimasi 378 menit (metode manual) menjadi 9 menit berkat kemampuan eksekusi paralel, menghasilkan tingkat efisiensi waktu mencapai 97,6%. Selain kecepatan, sistem ini juga berhasil menstandarisasi konfigurasi, mengeliminasi kesalahan teknis, dan mempermudah pemantauan infrastruktur secara terpusat.

1. Pendahuluan

Di era modern ini, meningkatnya ketergantungan terhadap teknologi informasi menjadikan jaringan komputer sebagai komponen utama dalam mendukung operasional bisnis dan interaksi pengguna[1]. Infrastruktur jaringan yang handal memungkinkan pertukaran informasi yang cepat, kolaborasi tim yang efisien, serta penerapan sistem keamanan terintegrasi[2]. Dalam infrastruktur tersebut, terdapat layanan penting yang mendukung kelancaran komunikasi data, yaitu *Domain Name System (DNS)* dan *Network Time Protocol (NTP). DNS* berperan sebagai basis data terdistribusi yang menerjemahkan nama host (hostname) menjadi alamat *IP* untuk identifikasi perangkat[3]. Sementara itu, NTP berfungsi mensinkronkan waktu secara presisi di seluruh perangkat jaringan. Pada industri ritel, akurasi waktu dari NTP sangat dibutuhkan untuk memastikan validitas pencatatan log transaksi penjualan (*Point of Sales*) dan sinkronisasi data rekaman keamanan (CCTV) antar cabang agar tetap konsisten dan akurat[4].

Penelitian ini dilaksanakan pada salah satu perusahaan ritel berskala nasional di Indonesia yang memiliki jaringan cabang yang tersebar luas. Setiap cabang dituntut untuk mengelola infrastruktur teknologi informasi (TI) secara mandiri, termasuk layanan *DNS* dan *NTP*, guna mendukung operasional harian. Berdasarkan studi dokumentasi awal, diketahui bahwa setiap cabang umumnya memiliki dua unit *server* (utama dan cadangan) yang memerlukan pemeliharaan rutin. Hingga saat ini, seluruh proses konfigurasi, pembaruan, dan manajemen *server* tersebut masih dilakukan dengan metode manual (konvensional).

Proses konfigurasi manual ini dinilai tidak efisien karena melibatkan tahapan yang panjang dan repetitif[5]. Teknisi harus menyunting berkas konfigurasi di direktori */etc/bind/,* mengatur sinkronisasi NTP, melakukan pembersihan *cache (flush cache),* hingga memulai ulang layanan (*restart service)* satu per satu. Berdasarkan wawancara dengan *IT Network Manager*, konfigurasi standar untuk satu *server* memakan waktu rata-rata 30 menit. Permasalahan ini semakin mendesak mengingat adanya kebijakan peremajaan infrastruktur *server (server refreshment*) untuk perangkat yang berusia lebih dari lima tahun. Pada tahun 2025, diproyeksikan terdapat puluhan *server* baru yang harus diinstalasi dari awal. Kondisi ini menciptakan beban kerja yang tidak seimbang bagi tim teknisi IT yang hanya beranggotakan dua orang. Ketidakseimbangan antara volume pekerjaan dan sumber daya manusia ini meningkatkan risiko terjadinya *human error* yang dapat berakibat fatal pada operasional toko dan *office*[6].

Untuk mengatasi permasalahan tersebut, diperlukan modernisasi manajemen *server* melalui pendekatan *Infrastructure as Code (IaC*). *IaC* merupakan pendekatan pengelolaan infrastruktur TI yang dilakukan menggunakan skrip atau kode otomatis, sehingga konfigurasi tidak lagi dilakukan secara manual. Pendekatan ini membantu mengurangi risiko *human error*, serta mendukung praktik *Continuous Integration/Continuous Deployment (CI/CD)* agar penerapan layanan TI menjadi lebih cepat dan terstandarisasi[7].

Solusi teknis yang diusulkan dalam penelitian ini adalah penggunaan *Ansible. Ansible* merupakan perangkat lunak *open-source* untuk otomatisasi konfigurasi yang bekerja secara *agentless* (tanpa agen)[8]. Keunggulan arsitektur *agentless* ini sangat cocok diterapkan pada topologi jaringan ritel dengan banyak cabang, karena tidak membebani sumber daya *server* di cabang dengan instalasi agen tambahan. Komunikasi dilakukan secara aman melalui protokol SSH, dan skrip konfigurasi ditulis dalam format YAML yang sederhana namun *powerful*[9].

Guna menyempurnakan manajemen otomatisasi tersebut, penelitian ini juga mengintegrasikan *Semaphore*, sebuah antarmuka berbasis *web (Web GUI)* untuk *Ansible*. *Semaphore* tidak hanya memudahkan administrator dalam menjalankan playbook melalui tampilan grafis, tetapi juga menyediakan fitur manajemen proyek dan *audit trail* (rekam jejak). Fitur ini sangat penting bagi perusahaan untuk memantau siapa yang melakukan perubahan konfigurasi dan kapan perubahan tersebut dilakukan, sehingga aspek keamanan dan akuntabilitas tetap terjaga. Integrasi *Ansible* dan

Semaphore diharapkan dapat mengubah manajemen *server* yang semula manual menjadi terotomatisasi, terpusat, efisien, dan mudah diaudit[10].

Beberapa penelitian sebelumnya telah membahas penerapan otomatisasi infrastruktur TI menggunakan pendekatan *IaC* berbasis *Ansible* dan *Semaphore* yang terbukti mampu meningkatkan efisiensi pengelolaan jaringan dan *server* , khususnya pada otomatisasi layanan tertentu seperti *VLAN* dan *DHCP* serta pada lingkungan *server* berbasis *container* dan *cloud*. Namun demikian, penelitian-penelitian tersebut belum secara spesifik mengkaji otomatisasi layanan DNS dan NTP secara terintegrasi. Selain itu, aspek pengelolaan terpusat pada lingkungan perusahaan ritel dengan banyak cabang yang tersebar secara geografis masih jarang dibahas dalam penelitian sebelumnya[7][8][10]. Oleh karena itu, penelitian ini bertujuan untuk merancang dan menerapkan sistem otomatisasi konfigurasi *server* dengan fokus pada layanan DNS dan NTP secara terpusat pada perusahaan *ritel multi-cabang*. Implementasi dilakukan dengan membangun *Ansible Controller* pada lingkungan *Virtual Machine (VM)* sebagai pusat kendali otomatisasi. Sistem yang dirancang diharapkan mampu meningkatkan efisiensi waktu konfigurasi, mengurangi risiko kesalahan konfigurasi *(human error)*, serta mendukung modernisasi infrastruktur TI pada perusahaan ritel.

2. Metode Penilitian

Metode penelitian merupakan komponen vital dalam sebuah studi ilmiah yang berfungsi sebagai landasan untuk menjawab rumusan masalah dan mencapai tujuan penelitian secara sistematis dan terukur[11]. Penelitian ini menerapkan metode eksperimental dengan pendekatan analisis komparatif (*comparative analysis*). Desain penelitian yang digunakan adalah *Pre-test and Post-test Design*, di mana pengukuran kinerja dilakukan pada objek penelitian yang sama (*server*) dalam dua kondisi berbeda.

Kondisi pertama (*pre-test*) adalah pengukuran kinerja saat manajemen *server* masih dilakukan secara manual (konvensional). Kondisi kedua (*post-test*) adalah pengukuran setelah penerapan otomatisasi menggunakan *Ansible* dan *Semaphore*. Metode ini dipilih karena kemampuannya memberikan data kuantitatif yang valid terkait perbandingan efisiensi waktu dan konsistensi konfigurasi sebelum dan sesudah sistem baru diterapkan[12].

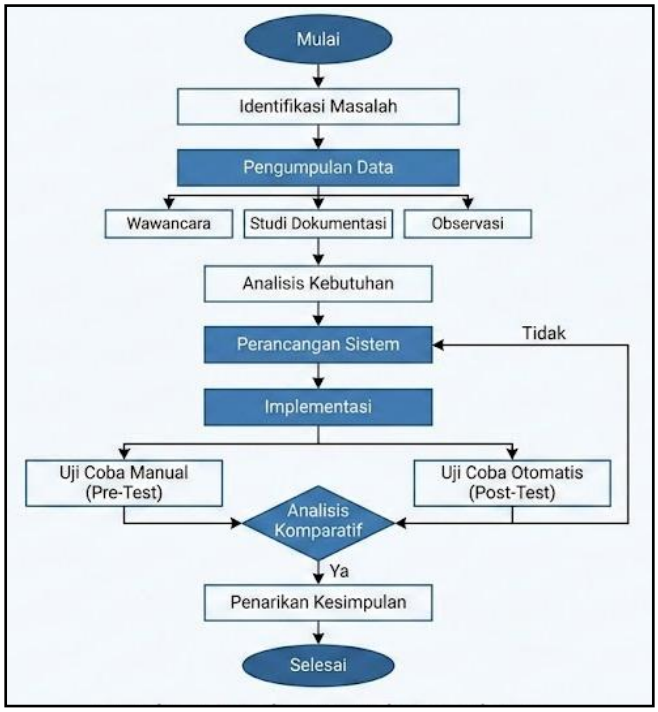
Analisis data dilakukan dengan membandingkan waktu konfigurasi *server* antara metode manual dan metode otomatisasi. Perhitungan efisiensi waktu digunakan sebagai indikator kinerja sistem, dengan rumus dan hasil perhitungannya disajikan pada bagian hasil dan pembahasan. Seluruh durasi pengujian dicatat dalam satuan menit dan dibulatkan ke menit terdekat untuk menyederhanakan proses analisis serta menjaga konsistensi perhitungan. Pendekatan ini bertujuan untuk memastikan evaluasi kinerja dilakukan secara kuantitatif dan dapat direplikasi.

Pengujian sistem dilakukan pada lingkungan *server* yang terdiri dari *server* fisik dan *virtual machine* dengan spesifikasi perangkat keras dan perangkat lunak yang seragam untuk setiap skenario pengujian. Rincian kebutuhan perangkat keras dan sistem operasi yang digunakan dalam penelitian ini disajikan pada Tabel 3.1. dan Tabel 3.2 Lingkungan tersebut mencakup *server* fisik sebagai *host virtualisasi, Ansible Controller*, serta *server* DNS dan NTP baik pada skenario *pre-test* maupun *post-test*.

2.1 Metode Pengumpulan Data

Untuk mendapatkan data yang akurat, proses pengumpulan data dilakukan melalui tiga teknik utama yaitu wawancara, studi dokumentasi, dan observasi. Wawancara mendalam dilakukan pada tanggal 5 Mei 2025 dengan *IT Network Manager* untuk memetakan alur kerja saat ini dan mengidentifikasi kendala utama dalam konfigurasi manual. Selanjutnya, studi dokumentasi dilakukan sepanjang periode Mei hingga September 2025 dengan menelaah arsip konfigurasi, topologi jaringan, serta kebijakan peremajaan *server* (*server refreshment*). Terakhir, observasi langsung dilakukan untuk mencatat metrik kinerja seperti durasi waktu konfigurasi dan frekuensi kesalahan yang terjadi pada

kedua skenario pengujian. Seluruh tahapan penelitian ini, mulai dari identifikasi masalah hingga penarikan kesimpulan, dirangkum dalam alur penelitian yang sistematis sebagaimana ditunjukkan pada Gambar 1.

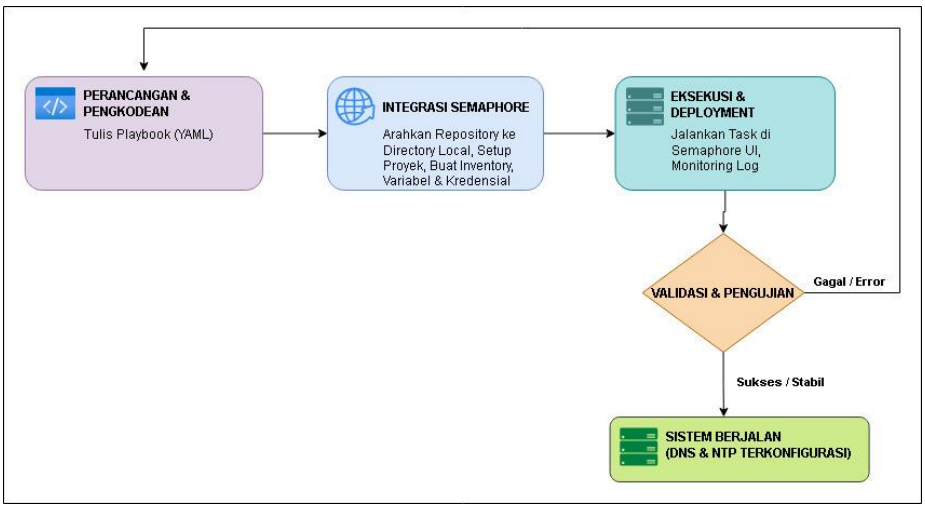


Gambar 1. Tahapan Metode Penelitian

2.2 Metode Pengembangan Sistem

Metode pengembangan sistem dalam penelitian ini mengadopsi pendekatan *Infrastructure as Code (IaC)*. *IaC* adalah paradigma modern di mana infrastruktur TI dikelola dan disediakan melalui kode skrip yang dapat dibaca mesin, bukan melalui konfigurasi perangkat keras fisik atau alat interaktif manual[13]. Pengembangan sistem dimulai dengan Tahap Desain, di mana arsitektur jaringan dirancang ulang untuk mendukung otomatisasi, dan variabel konfigurasi untuk layanan DNS serta NTP dipetakan agar dapat diterjemahkan ke dalam kode. Tahap ini menjadi cetak biru (*blueprint*) sebelum kode program ditulis.

Setelah desain matang, proses berlanjut ke Tahap Implementasi. Pada fase ini, rancangan diterjemahkan menjadi kode nyata berupa fail inventaris (*inventory file*) dan *Playbook Ansible* berformat YAML. Selain itu, dibuat pula *template* konfigurasi dinamis menggunakan Jinja2. Kode yang telah selesai ditulis kemudian diunggah ke dalam *repository* dan dihubungkan ke *Semaphore* sebagai pusat kendali. Tahap terakhir adalah Tahap *Deployment* dan Pengujian, di mana kode dieksekusi ke *server* target melalui *Semaphore*. Proses ini bersifat iteratif (berulang); jika ditemukan kesalahan (*error*) atau hasil konfigurasi tidak sesuai standar saat proses *deployment*, maka alur akan kembali ke tahap desain atau perbaikan kode hingga sistem berjalan stabil dan sesuai harapan, sebagaimana diilustrasikan pada Gambar 2.



Gambar 2. Bagan *IaC* berbasis *Ansible* dan *Semaphore*

3. Hasil dan Pembahasan

3.1 Hasil Identifikasi Kebutuhan Sistem

Tahapan awal dalam penelitian ini dimulai dengan identifikasi kebutuhan sistem yang komprehensif. Langkah ini bertujuan untuk memastikan bahwa lingkungan operasional, baik perangkat keras maupun perangkat lunak, mampu mendukung proses otomatisasi konfigurasi *server* menggunakan *Ansible* dan *Semaphore* secara optimal. Identifikasi kebutuhan ini dibagi menjadi dua kategori utama, yaitu kebutuhan perangkat keras (*hardware*) dan kebutuhan perangkat lunak (*software*).

Terkait kebutuhan perangkat keras, infrastruktur yang dibangun melibatkan tiga komponen utama: *server* fisik (*Host VM*) yang menjalankan virtualisasi *Proxmox, server virtual machine* sebagai *Ansible Controller*, dan *server* fisik target yang berfungsi sebagai node DNS dan NTP. Spesifikasi teknis yang digunakan, sebagaimana dirincikan pada Tabel 1, telah disesuaikan untuk menangani beban kerja manajemen konfigurasi secara terpusat. Sementara itu, kebutuhan perangkat lunak mencakup sistem operasi *Ubuntu Server* 22.04 LTS untuk pengendali, *Zentyal Server* 7.0 untuk *server* target, serta paket aplikasi pendukung seperti *Ansible* 2.10.8, *Semaphore*, dan layanan *Bind9*. Pemilihan perangkat lunak ini didasarkan pada kompatibilitas dan stabilitas sistem yang diperlukan dalam lingkungan produksi, seperti yang tertera pada Tabel 2.

Tabel 1. Kebutuhan Perangkat Keras

Jenis *Server*	Fungsi	Spesifikasi
Server Fisik (Host VM)	*Server* utama yang menjalankan virtual machine menggunakan Proxmox VE 7.2-3.	Type: Dell R820 CPU: 64 vCPU RAM: 540 GB Storage: 2.5 TB OS: Debian GNU/Linux Virtualisasi: Proxmox VE 7.2-3
Virtual Machine (*Ansible* Controller)	*Server* yang mengelola otomatisasi menggunakan *Ansible* dan *Semaphore*.	CPU: 8 vCPU RAM: 8 GB Storage: 300 GB OS: Ubuntu *Server* 22.04
Server Fisik (DNS dan NTP)	*Server* yang digunakan untuk layanan DNS dan NTP.	Type: Zyrex X302-RM CPU: 4 vCPU RAM: 16 GB Storage: 1 TB OS: Zentyal 7

Virtual Machine (DNS dan NTP)	*Server* yang digunakan untuk layanan DNS dan NTP pada tahap uji coba manual (pre-test).	CPU: 4 vCPU RAM: 4 GB Storage: 100 GB OS: Zentyal 7

Tabel 2. Kebutuhan Perangkat Lunak

Software	Fungsi
Ubuntu *Server* 22.04 LTS	Sistem operasi yang digunakan oleh *server Ansible* Controller.
Zentyal *Server* 7.0	Sistem operasi yang digunakan oleh *server* DNS dan NTP.
Ansible	*Tools* otomatisasi konfigurasi *server* .
Semaphore	Antarmuka berbasis *web* untuk mengelola tugas otomatisasi *Ansible*.
MariaDB	Digunakan sebagai basis data *Semaphore* untuk menyimpan data proyek, task, dan log eksekusi otomatisasi.
Bind9	Layanan DNS yang mendukung konfigurasi zona DNS primary dan secondary.
OpenSSH *Server*	Memungkinkan *Ansible* terhubung ke *server* lain dengan melalui koneksi SSH.

3.2 Hasil Pelaksanaan *Pretest* (Sebelum Otomatisasi)

Sebelum penerapan sistem otomatisasi, dilakukan pengujian awal (*pretest*) untuk memetakan kondisi operasional saat ini. Pada tahap ini, konfigurasi *server* DNS dan NTP dilakukan sepenuhnya secara manual sesuai prosedur standar yang berlaku di perusahaan. Proses ini mencakup penyuntingan berkas konfigurasi, pengaturan *IP Address*, hingga verifikasi layanan satu per satu. Berdasarkan pencatatan waktu yang disajikan pada Tabel 3, total durasi yang dibutuhkan untuk mengkonfigurasi satu unit *server* secara manual adalah 27 menit. Waktu ini tergolong lama karena banyaknya tahapan repetitif yang harus dilalui oleh teknisi.

Selain inefisiensi waktu, proses manual juga memiliki risiko teknis yang tinggi. Berdasarkan hasil observasi yang dirangkum pada Tabel 4, ditemukan beberapa kendala krusial seperti kesalahan penulisan format IP (*typo*), kesalahan penempatan direktori penyimpanan berkas konfigurasi, hingga kegagalan izin akses (*permission denied*) karena kelalaian penggunaan hak akses *root*. Kendala-kendala ini mengindikasikan bahwa metode manual tidak hanya lambat, tetapi juga rentan terhadap *human error* yang dapat mengganggu stabilitas layanan jaringan di cabang.

Tabel 3. Waktu (Durasi) Konfigurasi Manual

Kegiatan	Durasi (Menit)	Keterangan
Konfigurasi IP address dan DNS forwarder	2 menit	Melakukan konfigurasi IP address dan DNS forwarder pada *web* admin zentyal.
Proses konfigurasi *server* DNS dan NTP	20 menit	Meliputi semua pengaturan yang berhubungan dengan konfigurasi DNS dan NTP
Verifikasi hasil konfigurasi (uji ping, nslookup,ntp)	5 menit	Pengujian dilakukan dari client
Total	27 menit	-

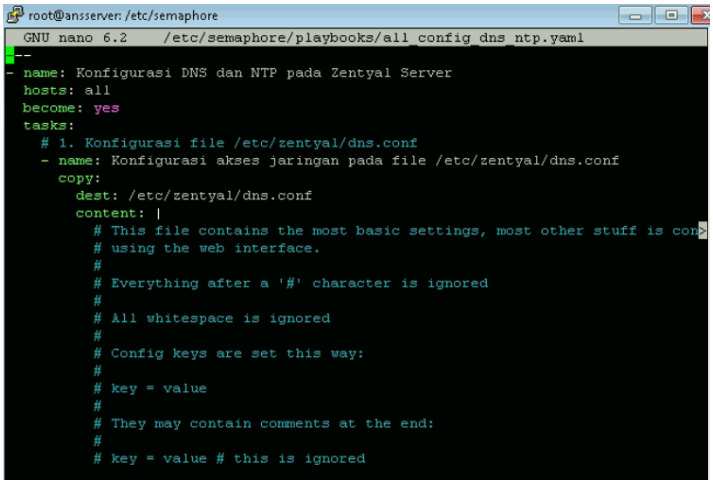
Tabel 4. Kendala Teknis Konfigurasi Manual

Tahapan Konfigurasi	Kemungkinan Kendala Teknis	Deskripsi Permasalahan
Konfigurasi akses jaringan pada file dns.conf	Kesalahan format IP/Subnet	Penulisan subnet tidak sesuai (contoh 192.168.0.0/8), menyebabkan client tidak dapat melakukan query DNS.

	IP address client yang didaftarkan tidak lengkap	Jika ada salah satu subnet/ip address client tidak ditambahkan, maka client yang ada pada jaringan tersebut tidak dapat melakukan query DNS.
	Tidak memiliki izin akses	File berada di /etc/zentyal/, hanya bisa diedit dengan sudo. Tanpa izin root, perubahan tidak dapat tersimpan.
	File tersimpan di direktori salah	Jika file disimpan di lokasi lain (contoh /home/user/), konfigurasi tidak dapat berjalan di sistem.
Replace file konfigurasi /etc/bind/ dan /usr/share/zentyal/stubs/dns/	Hak akses terbatas	File tidak bisa ditimpa tanpa hak root, jika mencobanya maka sistem menolak perubahan.
	File ditempatkan di direktori salah	Jika ditempatkan di direktori yang salah template konfigurasinya tidak termuat saat layanan DNS dijalankan.
Remove cache DNS	Salah hapus file cache	File cache berada di /var/cache/bind/; jika salah hapus, dapat menyebabkan gangguan layanan DNS hingga kegagalan sistem operasi.
Konfigurasi ntp.conf	Salah penulisan konfigurasi	Kesalahan pada penulisan *server* NTP time.google.com dapat membuat sinkronisasi waktu gagal.
	Tidak memiliki izin akses	File hanya bisa diubah dengan sudo; tanpa izin root, konfigurasi tidak tersimpan.
	File tersimpan di direktori salah	Jika file tersimpan di lokasi lain, konfigurasi NTP tidak dapat berjalan.

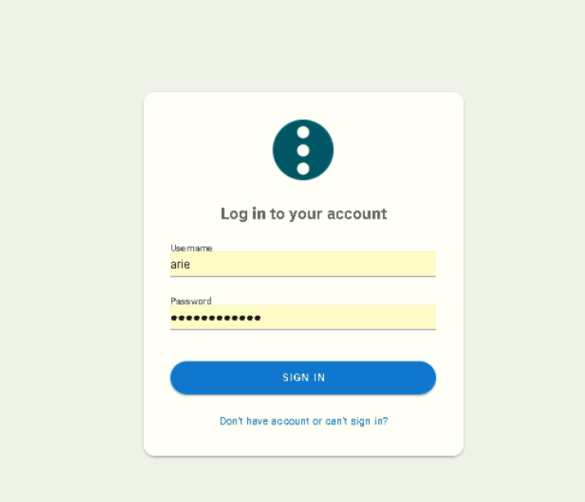
3.3 Hasil Implementasi Sistem

Menjawab permasalahan yang ditemukan pada tahap *pretest*, penelitian ini dilanjutkan dengan tahap implementasi sistem otomatisasi berbasis *Infrastructure as Code* (IaC). Implementasi dimulai dengan perancangan *Task Playbook Ansible*, yaitu skrip berformat YAML yang berisi instruksi terstruktur untuk konfigurasi DNS dan NTP (Gambar 3.1). Skrip ini kemudian diintegrasikan ke dalam *Semaphore* sebagai antarmuka manajemen terpusat.



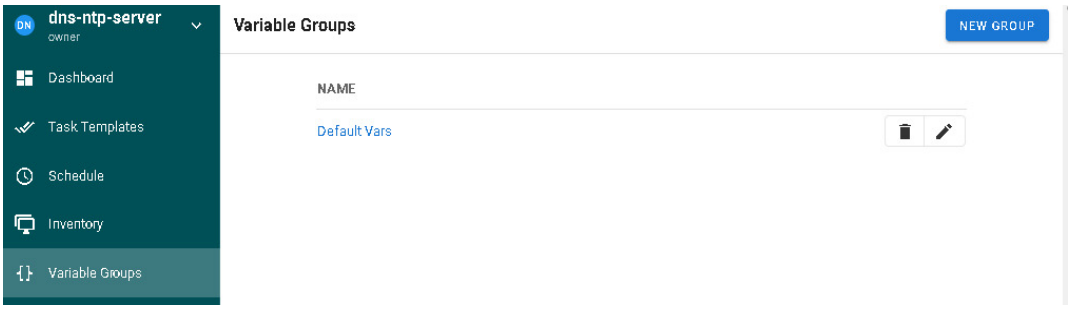
Gambar 3. Pembuatan ***Task Playbook Ansible***

Proses konfigurasi di *Semaphore* diawali dengan *login* administrator (Gambar 4), dilanjutkan dengan pembuatan *Key Store* untuk menyimpan kredensial SSH secara aman (Gambar 5). Selanjutnya, dilakukan pengelompokan variabel konfigurasi melalui menu *Variable Groups* (Gambar 6) dan penghubungan repositori kode (Gambar 7).



Gambar 4. Tampilan Halaman *Login Semaphore*

Gambar 5. Tampilan Halaman *Key Store*



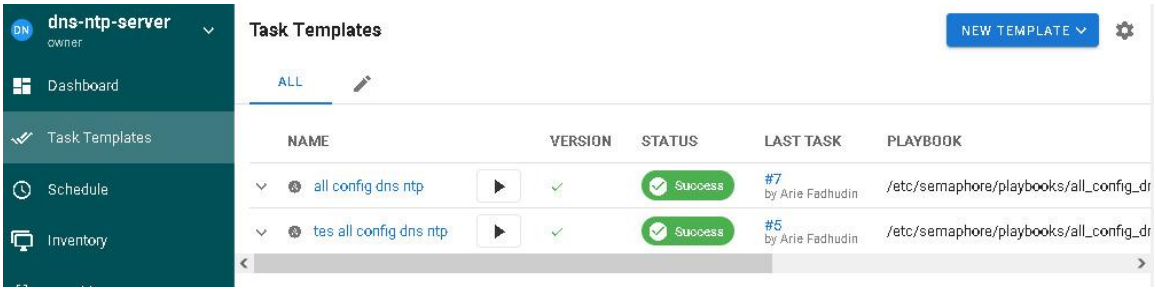
Gambar 6. Tampilan Halaman *Variable Groups*

Gambar 7. Tampilan Halaman Repositories

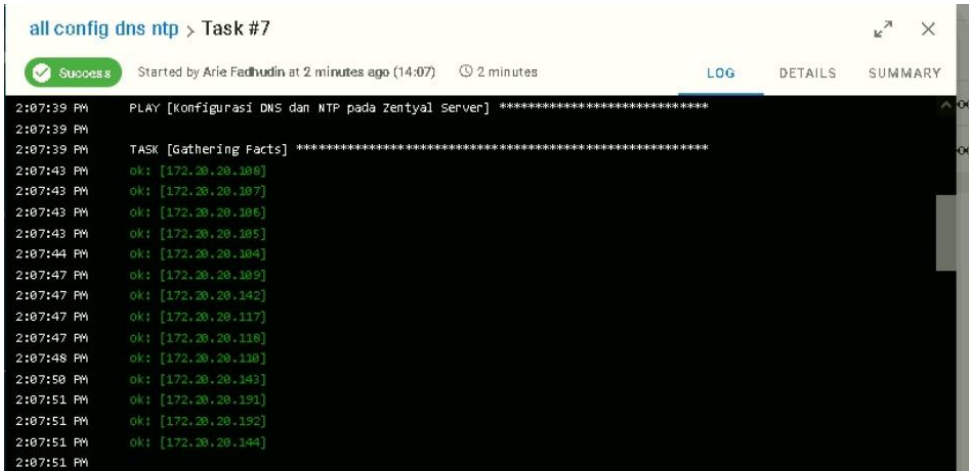
Server -server target didaftarkan ke dalam menu *Inventory* berdasarkan alamat IP masing-masing (Gambar 8). Tahap akhir dari implementasi adalah pembuatan *Task Template* (Gambar 9) yang

kemudian dieksekusi melalui fitur *Running Task*. Pada tahap ini, *Semaphore* mengeksekusi instruksi *Ansible* ke *server* target dan menampilkan log status keberhasilan secara *real-time* (Gambar 10).

Gambar 8. Tampilan Halaman *Inventory*



Gambar 9. Tampilan Halaman *Task Template*



Gambar 10. Tampilan *Running Task*

3.4 Hasil Pelaksanaan *Posttest* dan Pembahasan Efisiensi

Setelah sistem otomatisasi berhasil diterapkan, dilakukan pengujian akhir (*posttest*) untuk mengukur dampak peningkatan kinerja. Perbedaan mendasar antara metode manual dan otomatis terletak pada mekanisme eksekusinya; metode manual bekerja secara serial (satu per satu), sedangkan *Ansible* bekerja secara paralel (serentak).

Tabel 5. Perbandingan Waktu (Durasi) Konfigurasi

Pretest (1 *Server*)	Durasi (Menit)	Posttest (Otomatisasi 1 *Server*)	Durasi (Menit)	Posttest (Otomatisasi 14 *Server*)	Durasi (Menit)
Konfigurasi IP Address dan DNS Forwarder	2 menit	Konfigurasi IP Address dan DNS Forwarder	2 menit	Konfigurasi IP Address dan DNS Forwarder	2 menit

Proses konfigurasi *server* DNS dan NTP	20 menit	Proses konfigurasi *server* DNS dan NTP	1 menit	Proses konfigurasi *server* DNS dan NTP	2 menit
Verifikasi hasil konfigurasi (uji ping, nslookup, ntp)	5 menit	Verifikasi hasil konfigurasi (uji ping, nslookup, ntp)	5 menit	Verifikasi hasil konfigurasi (uji ping, nslookup, ntp)	5 menit
Total	27 menit	Total	8 menit	Total	9 menit

Berdasarkan data pada Tabel 5, waktu konfigurasi untuk satu *server* menggunakan otomatisasi turun drastis menjadi 8 menit, dibandingkan 27 menit pada metode manual. Peningkatan efisiensi menjadi jauh lebih signifikan ketika diuji pada skala yang lebih besar, yaitu 14 *server*. Secara manual, konfigurasi 14 *server* membutuhkan waktu kumulatif selama 378 menit (sekitar 6 jam 18 menit). Namun, dengan kemampuan eksekusi paralel *Ansible*, ke-14 *server* tersebut dapat dikonfigurasi secara bersamaan dalam waktu total yang hampir sama dengan konfigurasi satu *server*, yaitu 9 menit.

Perhitungan Efisiensi

Mengacu pada penelitian Fitriwati, dkk (2025)[14], efisiensi waktu dihitung menggunakan rumus:

$$Efisiensi = \frac{Waktu_{Otomatis} - Waktu_{manual}}{Waktu_{manual}} \times 100\%$$

Berdasarkan data di atas, analisis efisiensi dilakukan pada dua skenario:

Skenario 1 *Server*

$$Efisiensi = \frac{27 - 8}{27} \times 100\% = 70{,}4\%$$

Skenario 14 *Server* (Implementasi Skala Luas)

Pada metode manual, teknisi harus mengerjakan *server* satu per satu (serial). Maka total waktu untuk 14 *server* adalah 27 menit x 14 = 378 menit (6 jam 18 menit). Sedangkan dengan *Ansible*, karena eksekusi bersifat paralel, waktu yang dibutuhkan untuk 14 *server* tetap relatif sama dengan 1 *server* , yaitu sekitar 9 menit.

$$Efisiensi = \frac{378 - 9}{378} \times 100\% = 97{,}6\%$$

Hasil perhitungan menunjukkan bahwa untuk konfigurasi satu *server*, tingkat efisiensi waktu mencapai 70,4%. Sedangkan untuk implementasi skala luas (14 *server*), tingkat efisiensi melonjak hingga 97,6%. Pekerjaan yang sebelumnya memakan waktu hampir satu hari kerja (6 jam lebih), kini dapat diselesaikan dalam waktu kurang dari 10 menit. Data ini membuktikan secara empiris bahwa penerapan *Ansible* dan *Semaphore* mampu memangkas waktu operasional secara ekstrem dan menghilangkan faktor kelelahan teknisi yang menjadi penyebab utama kesalahan konfigurasi. Hal ini membuktikan bahwa implementasi *Infrastructure as Code* sangat efektif untuk lingkungan perusahaan ritel dengan banyak cabang.

3.5 Pembahasan

Analisis terhadap data hasil penelitian menunjukkan disparitas kinerja yang signifikan antara metode manajemen konfigurasi manual dan otomatisasi berbasis *Ansible*. Pada tahap pengujian awal (*pretest*) dengan skenario satu *server* , proses manual membutuhkan waktu 27 menit akibat banyaknya tahapan repetitif yang harus dilakukan teknisi, mulai dari penyuntingan berkas hingga verifikasi

layanan. Sebaliknya, penerapan otomatisasi pada tahap *posttest* mampu memangkas durasi tersebut menjadi 8 menit, yang menghasilkan efisiensi waktu sebesar 70,4%. Peningkatan kinerja ini menjadi jauh lebih drastis ketika diuji pada skala implementasi luas yang melibatkan 14 *server* , di mana efisiensi melonjak hingga 97,6%. Fenomena ini mengonfirmasi bahwa pendekatan otomatisasi tidak hanya mempercepat proses kerja, tetapi juga menawarkan skalabilitas yang tidak dapat dicapai oleh metode manual konvensional.

Faktor determinan utama di balik lonjakan efisiensi tersebut terletak pada perubahan mekanisme eksekusi dari serial menjadi paralel. Dalam metode manual, waktu pengerjaan bersifat linier dan akumulatif, di mana total durasi berbanding lurus dengan jumlah *server* yang dikelola. Hal ini terlihat dari proyeksi waktu manual untuk 14 *server* yang mencapai 378 menit atau setara dengan lebih dari enam jam kerja. Sebaliknya, arsitektur *agentless* pada *Ansible* memungkinkan *controller* mengirimkan instruksi konfigurasi ke seluruh *inventory server* secara serentak (*parallel execution*). Kemampuan ini menihilkan antrian proses, sehingga durasi yang dibutuhkan untuk mengkonfigurasi 14 *server* relatif sama dengan durasi untuk satu *server* . Temuan ini membuktikan bahwa hambatan operasional dalam proyek peremajaan *server* (*server refreshment*) dapat dieliminasi melalui adopsi teknologi ini, sehingga sumber daya manusia di bidang TI dapat dialihkan untuk menangani pekerjaan yang lebih strategis.

Selain aspek kecepatan, transformasi ke metode *Infrastructure as Code* (IaC) memberikan jaminan validitas dan konsistensi konfigurasi yang tidak dimiliki oleh metode manual. Observasi selama penelitian menyoroti tingginya kerentanan *human error* pada proses manual, seperti kesalahan pengetikan IP (*typo*) atau kesalahan direktori, yang berpotensi menyebabkan kegagalan layanan. Dengan *Ansible*, seluruh konfigurasi didefinisikan dalam kode *Playbook* yang bersifat idempoten, artinya kode tersebut dapat dijalankan berulang kali dengan hasil akhir yang selalu konsisten dan seragam di seluruh cabang. Hal ini secara efektif menstandarisasi pengaturan layanan vital seperti DNS dan NTP, menghilangkan variabilitas konfigurasi antar-teknisi, dan meningkatkan stabilitas sistem secara keseluruhan.

Lebih jauh lagi, integrasi *Semaphore* sebagai antarmuka manajemen berbasis *web* (*Web GUI*) memberikan nilai tambah dari sisi tata kelola TI (*IT Governance*). *Semaphore* mentransformasi proses eksekusi yang sebelumnya berbasis terminal dan terdesentralisasi menjadi terpusat pada satu dasbor. Fitur ini tidak hanya mempermudah pemantauan status konfigurasi secara *real-time*, tetapi juga menyediakan rekam jejak audit (*audit trails*) yang lengkap mengenai siapa, kapan, dan apa yang diubah pada sistem. Bagi lingkungan perusahaan ritel, kapabilitas ini sangat krusial untuk menjaga akuntabilitas dan keamanan infrastruktur. Secara implikatif, modernisasi ini berkontribusi langsung pada keberlangsungan bisnis, mengingat presisi konfigurasi NTP sangat vital untuk validitas log transaksi penjualan (*Point of Sales*), serta kecepatan pemulihan sistem (*disaster recovery*) yang kini dapat dilakukan dalam hitungan menit.

Hasil penelitian ini sejalan dengan sejumlah penelitian terdahulu yang mengkaji penerapan otomatisasi konfigurasi menggunakan pendekatan *Infrastructure as Code* dengan *tools* lain seperti *Chef* dan *Puppet*. Studi-studi tersebut menunjukkan bahwa otomatisasi mampu meningkatkan konsistensi konfigurasi dan mengurangi kesalahan manual, meskipun umumnya memerlukan kompleksitas awal yang lebih tinggi akibat penggunaan agen dan dependensi tambahan pada setiap node. Dibandingkan pendekatan tersebut, *Ansible* menawarkan keunggulan sebagai sistem otomatisasi yang bersifat *agentless* dan berbasis SSH, sehingga lebih mudah diimplementasikan pada lingkungan perusahaan ritel dengan infrastruktur yang heterogen[9][15]. Selain itu, integrasi *Ansible* dengan *Semaphore* memberikan nilai tambah berupa antarmuka manajemen terpusat dan mekanisme audit yang belum banyak dibahas dalam penelitian sebelumnya[10].

Meskipun menunjukkan peningkatan efisiensi yang signifikan, penelitian ini memiliki beberapa keterbatasan. Evaluasi kinerja difokuskan pada aspek efisiensi waktu konfigurasi dan belum mencakup analisis mendalam terhadap aspek non-fungsional seperti keamanan, skalabilitas, dan reliabilitas

jangka panjang. Dalam implementasi dunia nyata, eksekusi paralel *Ansible* berpotensi meningkatkan beban jaringan, terutama pada lingkungan dengan bandwidth terbatas. Selain itu, manajemen kunci SSH dan pengaturan hak akses pengguna pada *Semaphore* perlu dikelola secara hati-hati untuk mencegah risiko keamanan. Oleh karena itu, aspek-aspek tersebut menjadi perhatian penting dalam pengembangan dan penerapan sistem otomatisasi pada skala produksi.

4. Kesimpulan *(Ditulis dengan 11pt)*

Berdasarkan hasil penelitian dan analisis yang telah dilakukan, menunjukkan bahwa penerapan *Infrastructure as Code (IaC)* menggunakan *Ansible* dan *Semaphore* secara efektif meningkatkan efisiensi konfigurasi *server* DNS dan NTP, khususnya pada skala 14 *server*, dengan memangkas waktu konfigurasi dari 378 menit (manual) menjadi sekitar 9 menit (otomatisasi) atau setara efisiensi 97,6% karena eksekusi paralel *Ansible* lebih unggul dibanding proses manual yang bersifat serial. Selain mempercepat proses, otomatisasi melalui playbook terstandarisasi juga meningkatkan konsistensi konfigurasi antar *server* dan mengurangi human error, sementara *Semaphore* sebagai antarmuka manajemen berbasis *web* mendukung pengelolaan yang terpusat, terdokumentasi, dan mudah diawasi sehingga memperkuat tata kelola TI di lingkungan ritel multi-cabang.

Daftar Pustaka *(Ditulis dengan 11pt)*

[1] K. Rivaldi and G. Purnama, "Perancangan dan Penerapan Monitoring Infrastruktur Perangkat Jaringan Komputer pada Pusat Data dan Sarana Informatika melalui Pengaplikasian Zabbix Network Engineering," *Glob. Res. Innov. Edutech J.*, vol. 1, no. 1, pp. 250–256, 2025, [Online]. Available: https://e-journal.naureendigition.com/index.php/jam/article/download/1867/765

[2] A. N. Hasan and G. Purnama, "Perancangan dan Simulasi Jaringan Internet dengan Menerapkan Metode Pengembangan NDLC (Network Development Life Cycle) pada Akses Education Centre," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 8, no. 4, pp. 250–256, 2024, [Online]. Available: https://www.researchgate.net/publication/381643450_PERANCANGAN_DAN_SIMULASI_JARINGAN_INTERNET_DENGAN_MENERAPKAN_METODE_PENGEMBANGAN_NDLC_NETWORK_DEVELOPMENT_LIFE_CYCLE_PADA_AKSES_EDUCATION_CENTRE

[3] A. Mustofa and D. Ramayanti, "Implementasi Load Balancing dan Failover to Device Mikrotik Router Menggunakan Metode NTH (Studi Kasus: PT. GO-JEK Indonesia)," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 7, no. 1, pp. 139–144, 2020, doi: 10.25126/jtiik.202071638.

[4] A. B. Setiawan, "Implementasi Sinkronisasi Waktu dengan Network Time Protocol untuk Pemantauan Keamanan Aktivitas Jaringan Telekomunikasi," *J. Penelit. Pos dan Inform.*, vol. 10, no. 1, pp. 1–12, 2020, [Online]. Available: https://jppi.komdigi.go.id/index.php/jppi/article/view/93

[5] A. I. Ramdhani, Z. M. Subekti, I. Jaya, E. M. Putro, and A. Ramadhan, "Automasi Konfigurasi *Web* Service pada Ubuntu *Server* Menggunakan *Ansible* Berbasis Python," *DEVICE*, vol. 13, no. 1, pp. 88–99, 2023, [Online]. Available: https://garuda.kemdiktisaintek.go.id/documents/detail/3532751

[6] N. I. of S. and Technology, "Security and Privacy Controls for Information Systems and Organizations," NIST, 2020. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf

[7] I. Wayan, I. M. Piarsa, and P. Buana, "Integrasi Infrastructure as Code dengan Continuous Integration/Continuous Deployment di Google Cloud Platform," *J. Teknol. Inf. dan Komput.*, vol. 7, no. 1, pp. 45–54, 2024, [Online]. Available: https://ejurnal.umri.ac.id/index.php/JIK/article/view/7236?utm_source=chatgpt.com

[8] Y. A. Chandrawaty and I. P. Hariyadi, "Implementasi *Ansible* Playbook untuk Mengotomatisasi Manajemen Konfigurasi VLAN Berbasis VTP dan Layanan DHCP," *J. Bumigora Inf. Technol.*, vol. 3, no. 2, pp. 107–122, 2021, [Online]. Available: https://journal.universitasbumigora.ac.id/index.php/bite/article/view/1577/832

[9] H. Katariya, M. Gedam, R. Lolage, S. Patil, and U. Shrivastav, "Comparative Analysis of Configuration Management Tools: Chef vs. *Ansible*, SaltStack, and Puppet," 2025. [Online]. Available: https://easychair.org/publications/preprint/sbfH

[10] P. U. Gunadarma, "Otomatisasi *Server* Menggunakan *Ansible Semaphore* dengan Linux Debian Buster pada Docker Container," *Repos. Univ. Gunadarma*, 2021, [Online]. Available: https://library.gunadarma.ac.id/repository/otomatisasi-*server* -menggunakan-*Ansible*-*Semaphore*-dengan-linux-debian-buster-pada-docker-container-penelitian

[11] W. I. Yanti and D. Ramayanti, "Pengembangan Aplikasi Posyandu Berbasis *Web* dalam Mendukung Monitoring Gizi Balita untuk Pencegahan Stunting (Studi Kasus: Posyandu Pamuji Rahayu)," *RJTI Riau J. Tek. Inform.*, vol. 4, no. 2, pp. 180–187, 2025, doi: 10.30606/rjti.v4i2.3409.

[12] W. William and H. Hita, "Mengukur Tingkat Pemahaman Pelatihan PowerPoint Menggunakan Quasi-Experiment One-Group Pretest-Posttest," *J. Sifo Mikroskil*, vol. 20, no. 1, p. 73, 2019, doi: 10.55601/jsm.v20i1.650.

[13] I. P. A. E. Pratama and P. B. S. W. Putra, "Pengujian *IaC* Berbasis DevOps dan *Ansible* Menggunakan Metode Black Box Testing," *Fakt. Exacta*, vol. 15, no. 2, pp. 84–91, 2022, doi: 10.30998/faktorexacta.v15i2.12039.

[14] F. Sovia, R. Al-Adawiyah, and T. P. Yuliana, "Evaluation of the Effectiveness of Renggak Leaf Extract (Amomum dealbatum Roxb.) as an Antihyperuricemic in Mice," *J. Pharm. Sci.*, vol. 7, no. 1, pp. 1082–1090, 2025, doi: 10.36490/journal-jps.com.v7i1.747.

[15] V. Aditi, "Optimizing Infrastructure Management using *Ansible*," *Int. J. Inf. Technol.*, vol. 11, no. 5, 2025, [Online]. Available: https://www.ijitjournal.org/volume-11/issue-5/IJIT-V11I5P2.pdf