Jurnal Teknik Informatika dan Teknologi Informasi
Volume 6 Nomor 1 April 2026
E-ISSN .: 2827-9387, P-ISSN .: 2827-9379, Hal. 603-616
DOI: https://doi.org/10.55606/jutiti.v6i1.7107
Tersedia: https://journalshub.org/index.php/jutiti

Rekayasa Otomatisasi Infrastruktur Jaringan Virtual Berbasis Open Vswitch Menggunakan Ansible pada Proxmox VE

Muhammad Wardhani[1*], Muhammad Irfan[2], Hidayatunnisa'i[3], Fidyah Ajeng Wulandari[4]
[1-4]Universitas Mbojo Bima, Indonesia
**Penulis Korespondensi: Muhammadwardhani1807@gmail.com*

Abstract. *Proxmox provides facilities for building private networks that enable each host to share physical resources through virtual networking, including Virtual Local Area Networks (VLANs). The Proxmox server supports Open vSwitch as a virtual switch, which is widely used among cloud developers to manage traffic between virtual machines and external communications. This study employs the Network Development Life Cycle (NDLC) method and the implementation of an automation engine based on Ansible to enhance the efficiency of virtual network configuration and reduce human errors. The testing results, based on five experimental scenarios, indicate that the automation system is capable of accelerating network creation operations by an average of 7 minutes and 43 seconds, network addition operations by an average of 7 minutes and 16 seconds, and network deletion operations by an average of 39 seconds compared to manual methods. These findings demonstrate that virtual network automation in Proxmox VE can significantly improve productivity and reliability in network management.*

Keywords: *Ansible; Network Development Life Cycle (NDLC); Open Vswitch; Proxmox VE; Virtual Network Automation.*

Abstrak. Proxmox menyediakan fasilitas untuk membangun jaringan privat yang memungkinkan setiap host berbagi sumber daya fisik melalui jaringan virtual, termasuk Virtual Local Area Network (VLAN). Server Proxmox mendukung Open vSwitch sebagai switch virtual yang banyak digunakan oleh pengembang cloud untuk mengelola lalu lintas jaringan antara mesin virtual dan komunikasi eksternal. Pengelolaan jaringan virtual secara manual dapat menyebabkan proses konfigurasi menjadi lebih kompleks serta meningkatkan kemungkinan terjadinya kesalahan manusia. Penelitian ini bertujuan untuk menerapkan otomatisasi infrastruktur jaringan virtual berbasis Ansible pada Proxmox VE guna meningkatkan efisiensi konfigurasi dan pengelolaan jaringan. Metode yang digunakan dalam penelitian ini adalah Network Development Life Cycle (NDLC) dengan penerapan automation engine berbasis Ansible sebagai alat untuk melakukan konfigurasi jaringan secara otomatis. Hasil pengujian berdasarkan lima skenario eksperimen menunjukkan bahwa sistem otomatisasi mampu mempercepat operasi pembuatan jaringan dengan rata-rata waktu 7 menit 43 detik, operasi penambahan jaringan dengan rata-rata waktu 7 menit 16 detik, serta operasi penghapusan jaringan dengan rata-rata waktu 39 detik dibandingkan metode manual. Hasil penelitian menunjukkan bahwa penerapan otomatisasi jaringan virtual pada Proxmox VE dapat meningkatkan produktivitas, efisiensi, dan keandalan dalam manajemen infrastruktur jaringan.

Kata Kunci: Ansible; Open Vswitch;Otomatisasi Jaringan Virtual; Proxmox VE; Siklus Hidup Pengembangan Jaringan (NDLC).

1. LATAR BELAKANG

Teknologi virtualisasi telah menjadi tren sekaligus kebutuhan dalam pengelolaan infrastruktur teknologi informasi. Banyak perusahaan memanfaatkan virtualisasi pada server untuk meningkatkan efisiensi penggunaan sumber daya, khususnya dalam mengoptimalkan kinerja prosesor multi-core serta mengurangi pemborosan daya komputasi yang mahal. Selain itu, penggunaan virtualisasi juga berdampak pada penghematan biaya listrik karena operasional dapat dijalankan hanya dengan satu atau beberapa server fisik saja. Syaifuddin et al., (2025) (Pitra et al., 2025). Perangkat lunak yang digunakan untuk mengelola virtualisasi dikenal sebagai hypervisor, Sistem Operasi Virtual yang paling relevan digunakan adalah Proxmox

Naskah Masuk: 20 Februari 2026; Revisi: 22 Maret 2026; Diterima: 28 April 2026; Terbit: 30 April 2026

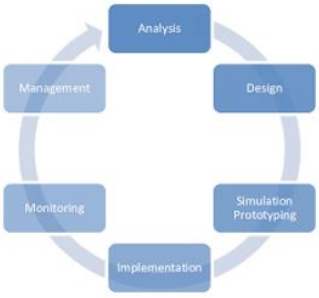
Virtual Environment (Proxmox VE). Proxmox VE merupakan sistem operasi virtual berbasis open-source yang populer di kalangan pengguna virtualisasi. Platform ini menyediakan berbagai fitur, termasuk dukungan pembuatan dan pengelolaan *virtual machine* (VM) (Pramadani et al., 2025). Setiap VM berjalan pada satu host fisik sehingga berbagi sumber daya perangkat keras, termasuk konektivitas jaringan.Virtual networking memungkinkan pembentukan beberapa jaringan virtual di atas satu infrastruktur jaringan fisik. Melalui pendekatan ini, administrator dapat membuat dan mengelola berbagai jaringan virtual secara terpisah pada tingkat perangkat lunak tanpa saling mengganggu, serta memungkinkan keberadaan beberapa jaringan virtual yang berbeda dalam satu lingkungan fisik yang sama. Dalam implementasinya, virtual switch (vSwitch) berfungsi menghubungkan Network *Interface Controller* (NIC) dengan VM untuk menjamin konektivitas jaringan dan mendukung topologi mesin virtual (Gde, et al., 2025). Proxmox mendukung penggunaan Open vSwitch (OVS) sebagai virtual switch. OVS banyak digunakan dalam lingkungan komputasi awan karena kemampuannya dalam mengatur lalu lintas data antar VM maupun komunikasi dengan jaringan eksternal (Intelligence et al., 2025; Grantor, 2026).

Virtualisasi jaringan dapat diterapkan dalam berbagai bentuk, salah satunya melalui Virtual Local Area Network (VLAN). VLAN memungkinkan pembentukan jaringan logis yang terpisah dari struktur fisik jaringan. Pada perangkat switch yang dapat dikelola (manageable switch), konfigurasi VLAN dapat dilakukan dengan batas maksimal 4096 VLAN (Widjajarto et al., 2025). Jumlah ini relatif besar untuk kebutuhan umum, namun dapat menjadi kendala bagi penyedia layanan internet (ISP) atau penyedia cloud berskala global dengan jumlah pelanggan yang sangat banyak. Untuk mengatasi keterbatasan tersebut, digunakan teknologi Virtual eXtensible Local Area Network (VXLAN) (Emmungil, 2025). VXLAN memiliki identitas segmen 24-bit yang disebut Virtual Network Identifier (VNI), sehingga memungkinkan pembentukan hingga sekitar 16 juta segmen jaringan virtual dalam satu domain administratif. (Zulfikar et al., 2025). Pengelolaan jaringan virtual berbasis Open vSwitch, termasuk konfigurasi VLAN dan VXLAN, memerlukan ketelitian tinggi dan berpotensi menimbulkan kesalahan manusia. Tantangan ini semakin kompleks bagi penyedia layanan cloud dengan jumlah pelanggan yang terus meningkat, karena konfigurasi perangkat jaringan virtual harus disesuaikan dengan pertambahan penyewaan server cloud (Vitor, 2025). Jika dilakukan secara manual dan berulang, proses tersebut menjadi tidak efisien, memakan waktu lama, serta meningkatkan risiko kesalahan konfigurasi (Anjani et al.,, 2024; Putra, 2025). Untuk mengatasi permasalahan tersebut, diperlukan solusi yang mampu mempercepat

proses implementasi sekaligus meminimalkan kesalahan akibat konfigurasi manual. Salah satu pendekatan yang dapat diterapkan adalah manajemen konfigurasi menggunakan Ansible. Otomatisasi memungkinkan pekerjaan dilakukan dan dikelola secara sistematis oleh sistem tanpa pengawasan manusia secara terus-menerus. Ansible merupakan mesin otomatisasi teknologi informasi yang sederhana dan fleksibel, yang dapat digunakan untuk provisioning cloud, manajemen konfigurasi, deployment aplikasi, orkestrasi layanan, serta kebutuhan TI lainnya (Minahasa, 2024). Dalam penelitian ini, otomatisasi Ansible diwujudkan dalam bentuk playbook yang berisi serangkaian tugas terkait pengelolaan virtual networking, khususnya VLAN dan VXLAN. Pengujian dilakukan untuk memastikan fungsi playbook berjalan sesuai dengan kebutuhan sistem. Penelitian ini bertujuan untuk menawarkan solusi konfigurasi jaringan virtual berbasis Open vSwitch pada Proxmox VE agar lebih terstruktur, efisien, dan mudah dikelola, sekaligus memberikan pemahaman mengenai implementasi otomatisasi jaringan virtual pada lingkungan tersebut. (Artikel et al., 2025; Wardhani et al., 2023).

2. METODE PENELITIAN

Penelitian ini menerapkan metode Network Development Life Cycle (NDLC) sebagai kerangka kerja dalam pelaksanaannya. Meskipun NDLC terdiri dari enam tahapan, penelitian ini hanya mengimplementasikan tiga tahapan inti, yaitu analisis kebutuhan, perancangan sistem, serta simulasi dan pembuatan prototipe. Ketiga tahapan tersebut digambarkan pada Gambar 1.



Gambar 1. Metode NDLC.

Analisis

Pada tahap analisis, peneliti melakukan pengumpulan data melalui kajian pustaka terhadap berbagai referensi yang relevan, seperti jurnal yang membahas Open vSwitch serta implementasi Virtual Local Area Network (VLAN) pada lingkungan mesin virtual. Selain itu, data pendukung juga diperoleh dari beragam sumber lain, antara lain buku, makalah, e-book, dan artikel ilmiah. Hasil kajian menunjukkan bahwa penelitian sebelumnya mengenai Open

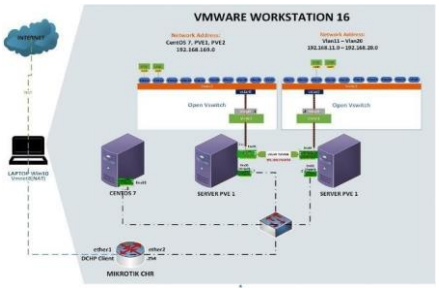
vSwitch umumnya berfokus pada penerapan dan evaluasi kinerja, namun belum diterapkan pada server Proxmox. Implementasi sebelumnya dilakukan secara manual dan belum memanfaatkan alat otomatisasi seperti Ansible. Sementara itu, penelitian terkait Ansible lebih banyak membahas manajemen konfigurasi pada server cloud, dan belum diterapkan khusus pada Proxmox VE. Berdasarkan hal tersebut, penelitian ini menitikberatkan pada penerapan Open vSwitch pada server Proxmox serta otomatisasi manajemen konfigurasi menggunakan Ansible, dengan tujuan meningkatkan efisiensi, konsistensi, dan akurasi dalam pengelolaan jaringan virtual.

Desain

Tahap ini meliputi perancangan yang mencakup desain topologi jaringan, konfigurasi antarmuka pada server PVE1 dan PVE2, perancangan keamanan IP, desain sistem otomatisasi, serta penentuan kebutuhan perangkat keras dan perangkat lunak.

Desain Topologi Jaringan

Desain Topologi jaringan yang digunakan dalam penelitian ini ditampilkan pada Gambar 2.

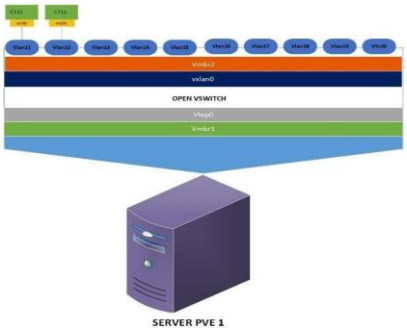


Gambar 2. Rancangan Topologi Jaringan.

Desain Topologi Jaringan disimulasikan menggunakan virtualisasi pada satu komputer dengan sistem operasi Windows 10 yang menjalankan hypervisor VMware Workstation 16. Pada VMware Workstation dibuat empat *Virtual machine* (VM), yaitu CentOS 7, server Proxmox Virtual Environment 1 (PVE1), server PVE2, dan Mikrotik CHR. CentOS 7 berfungsi sebagai server Ansible yang menjadi mesin otomatisasi, sedangkan PVE1 dan PVE2 menjadi target pengelolaan konfigurasi jaringan virtual berbasis Open vSwitch (OVS). Mikrotik CHR digunakan sebagai gateway Internet untuk menghubungkan ketiga VM tersebut ke jaringan eksternal.

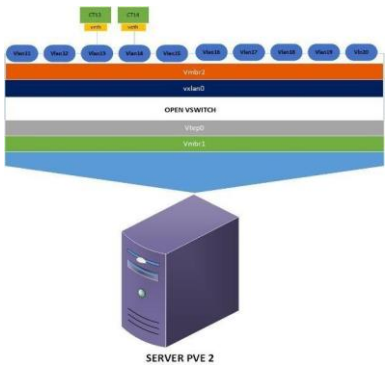
Perancangan Antarmuka Jaringan Virtual pada Server PVE1 dan PVE2

Perancangan antarmuka jaringan virtual akan diterapkan pada server PVE1, seperti ditunjukkan pada Gambar 3.



Gambar 3. Perancangan Antarmuka Jaringan Virtual pada Server PVE1.

Pada perancangan antarmuka jaringan virtual di server PVE1, terdapat beberapa komponen utama. Interface vmbr1 terhubung langsung ke interface fisik server PVE1, yaitu ens33. Interface vtep0 berperan sebagai endpoint dari jaringan overlay atau jalur jaringan virtual, sedangkan vxlan0 berfungsi sebagai interface tunneling untuk komunikasi antar-VLAN pada server yang berbeda. Interface vmbr2 menghubungkan VLAN11 dengan VLAN20, di mana VLAN11 dan VLAN20 terhubung ke container CT11 dan CT12 pada server PVE1. Sementara itu, perancangan antarmuka jaringan virtual pada server PVE2 ditampilkan pada Gambar 4.

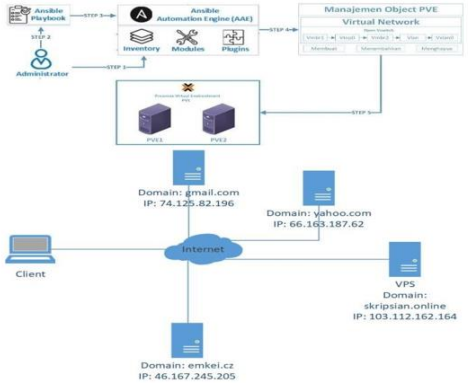


Gambar 4. Perancangan Antarmuka Jaringan Virtual pada Server PVE2.

Pada perancangan antarmuka jaringan virtual di server PVE2, beberapa komponen utama digunakan. Interface vmbr1 terhubung langsung ke interface fisik server PVE2, yaitu ens33. Interface vtep0 berfungsi sebagai endpoint jaringan overlay atau jalur jaringan virtual, sedangkan vxlan0 digunakan sebagai interface tunneling untuk komunikasi antar-VLAN pada server yang berbeda. Interface vmbr2 menghubungkan VLAN11 dengan VLAN20, di mana VLAN11 dan VLAN20 terhubung ke container CT13 dan CT14 yang berada pada server PVE2.

Perancangan Sistem Otomatisasi Jaringan Virtual

Perancangan sistem otomatisasi jaringan virtual berbasis Open vSwitch (OVS) akan diterapkan pada masing-masing server PVE menggunakan Ansible, seperti ditunjukkan pada Gambar 5.



Gambar 5. Perancangan Sistem Otomatisasi.

Perancangan sistem otomatisasi jaringan virtual terdiri dari lima tahap. Tahap pertama adalah penentuan *inventory* oleh Administrator pada CentOS 7, yang mencakup mesin target untuk otomatisasi, yaitu server PVE1 dan PVE2. Tahap kedua meliputi pembuatan file *playbook* dalam format YAML yang berisi tugas-tugas sesuai skenario pengujian, yaitu pembuatan, penambahan, dan penghapusan jaringan virtual berbasis Open vSwitch (OVS) pada setiap server PVE. Tahap ketiga adalah eksekusi playbook menggunakan *Ansible Automation Engine (AAE).* Hasil eksekusi tahap ketiga akan memengaruhi tahap keempat dan kelima, yang mencakup pengelolaan objek jaringan virtual berbasis OVS pada masing-masing server PVE.

Perancangan Alamat IP

Dalam perancangan alamat IP untuk jaringan virtual percobaan otomatisasi pada server PVE1 dan PVE2, digunakan empat alamat jaringan kelas C, yaitu 192.168.169.0/24, 192.168.11.0/24, 192.168.12.0/24, dan 192.168.13.0/24. Informasi rinci mengenai alokasi alamat IP untuk setiap mesin virtual dan klien Windows 10 disajikan pada Tabel 1.

Tabel 1. Perancangan Alamat IP.

No.	Nama Virtual Machine	Interface	IP Address	Subnet Mask
1	CentOS 7	Ens33	192.168.161.2	255.255.254.0
2	Pve1	Ens32	192.168.169.1	255.255.254.0
3	Pve2	Ens32	192.168.169.2	255.255.254.0
4	Mikrotik CHR	Ether2	192.168.169.254	255.255.254.0
5	Container 11	veth	192.168.11.100	255.255.254.0
6	Container 12	veth	192.168.12.112	255.255.254.0
7	Container 13	veth	192.168.13.113	255.255.254.0
8	Container 14	veth	192.168.14.114	255.255.254.0

No.	Nama Virtual Machine	Interface	IP Address	Subnet Mask
9	Client Win 10	Vmnet1	192.168.169.253	255.255.254.0

Kebutuhan Perangkat Keras dan Perangkat Lunak

Penelitian ini menggunakan satu laptop dengan spesifikasi prosesor Intel Core i3-6006U 2.00 GHz, SSD 250 GB, hard drive 1 TB, memori 12 GB, dan sistem operasi Windows 10 sebagai perangkat keras utama. Pada laptop ini diinstal VMware Workstation 16 untuk menjalankan empat *Virtual machine* (VM), yaitu server PVE1, server PVE2, server Ansible, dan Mikrotik CHR. Spesifikasi masing-masing VM adalah sebagai berikut: server PVE1 dan PVE2 masing-masing memiliki 1 core prosesor, 40 GB hard drive, dan memori 3 GB; server Ansible memiliki 1 core prosesor, 20 GB hard drive, dan memori 2 GB; sedangkan Mikrotik CHR menggunakan 1 core prosesor, 64 MB hard drive, dan memori 256 MB.

Dari sisi perangkat lunak, server PVE1 dan PVE2 menjalankan Proxmox Virtual Environment (Proxmox VE) sebagai sistem operasi utama, sedangkan server Ansible menggunakan CentOS 7 sebagai mesin otomatisasi. Selain itu, Linux Container Template CentOS 7 digunakan sebagai sistem operasi pada container yang berjalan di PVE1 dan PVE2. VMware Workstation 16 digunakan untuk menjalankan semua VM, termasuk CentOS 7, Mikrotik CHR, PVE1, dan PVE2. Mikrotik CHR berfungsi sebagai gateway Internet agar VM dapat terhubung ke jaringan eksternal. Untuk manajemen server, digunakan PuTTY untuk remote dan Google Chrome sebagai browser untuk mengakses GUI web dashboard server PVE.

Simulasi Prototype

Pada tahap ini, akan dilakukan percobaan simulasi yang meliputi proses instalasi dan konfigurasi pada setiap mesin virtual, sekaligus pengujian berbagai skenario untuk mengevaluasi kinerja sistem yang sedang dikembangkan.

Tahap Instalasi dan Konfigurasi

Pada tahap ini, dilakukan proses instalasi dan konfigurasi untuk setiap mesin virtual sesuai dengan desain percobaan. Pada server PVE1 dan PVE2, tahap instalasi dan konfigurasi mencakup penetapan alamat IP serta pemasangan paket-paket seperti Open vSwitch, Python-pip, Proxmoxer, dan ifupdown2. Sedangkan pada server CentOS 7, proses instalasi dan konfigurasi meliputi penetapan alamat IP, pemasangan paket seperti Epel-release, Ansible, Jinja2, serta pembuatan playbook Ansible. Untuk sisi klien, konfigurasi meliputi pengaturan alamat IP agar klien dapat mengakses antarmuka web berbasis Graphical User Interface (GUI) dari server PVE.

Skenario Percobaan

Pada tahap ini, ditampilkan hasil pengujian instalasi dan konfigurasi dengan menggunakan beberapa skenario percobaan, yang meliputi pengujian seperti pembuatan, penambahan, dan penghapusan jaringan virtual secara manual maupun otomatis. Selain itu, dilakukan penghitungan waktu yang dibutuhkan untuk pembuatan jaringan virtual secara manual dan otomatis. Rincian skenario percobaan ditunjukkan pada Tabel 2.

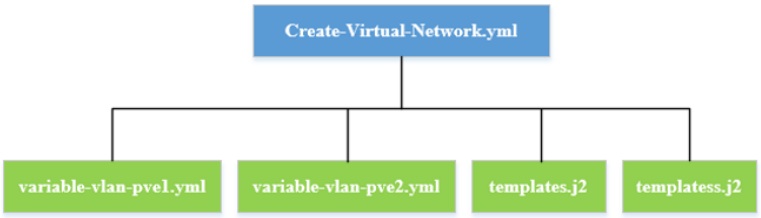
Tabel 2. Skenario Percobaan.

No.	Trials	Process	Description
1	Lima percobaan terkait pembuatan, penambahan, dan penghapusan jaringan virtual pada server PVE1 dan PVE2	Manual (melalui antarmuka web PVE1 dan PVE2)	Menghitung waktu minimum, maksimum, dan rata-rata untuk proses pembuatan, penambahan, dan penghapusan jaringan virtual.
2	Lima percobaan terkait pembuatan, penambahan, dan penghapusan jaringan virtual pada server PVE1 dan PVE2	Otomatisasi (dibuat menggunakan Ansible)	Menghitung waktu minimum, maksimum, dan rata-rata untuk proses pembuatan, penambahan, dan penghapusan jaringan virtual.

3. HASIL DAN PEMBAHASAN

Struktur Playbook Ansible dalam Pembuatan Jaringan Virtual

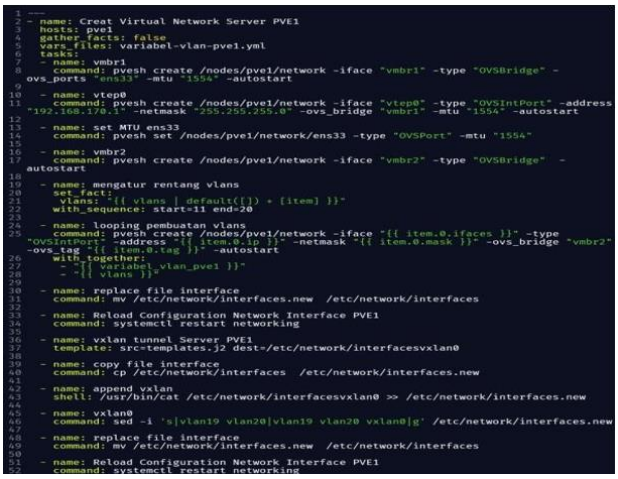
Desain sistem yang dibuat pada tahap perancangan dituangkan dalam bentuk *Playbook Ansible*. Terdapat tiga struktur playbook Ansible yang dihasilkan untuk mengotomatisasi pengelolaan konfigurasi jaringan virtual berbasis OVS, yaitu playbook untuk proses pembuatan (inisialisasi) jaringan virtual, playbook untuk memperbarui (*Update*) jaringan virtual, dan playbook untuk menghapus jaringan virtual. Struktur playbook Ansible untuk pembuatan jaringan virtual ditunjukkan pada Gambar 6.



Gambar 6. Struktur Playbook Ansible dalam Pembuatan Jaringan Virtual.

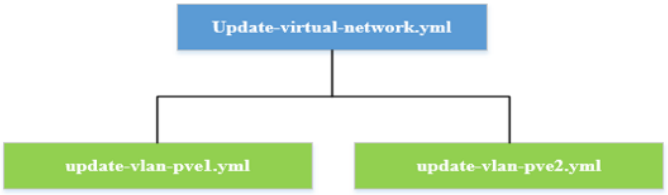
File *Create-Virtual-Network.yml* merupakan *playbook Ansible* yang memuat tugas-tugas terkait pembuatan jembatan *(bridge) bertipe OVSBridge* pada setiap *server PVE* dengan nama vmbr1 dan vmbr2, serta *interface* vtep0 bertipe OVSIntPort termasuk pengaturan alamatIP pada interface tersebut. Selain itu, playbook ini juga memuat tugas untuk membuat 10 (sepuluh) VLAN dengan ID 11 hingga 20, beserta pengaturan alamat IP pada *interface* yang datanya diambil dari file *variable-VLAN-pve1.yml* dan *variable-VLAN-pve2.yml*. Selanjutnya,

playbook ini membuat interface bertipe VXLAN (OVSTunnel) pada masing-masing server PVE untuk membentuk terowongan, sehingga host pada VLAN yang sama namun berada di server PVE berbeda dapat tetap saling berkomunikasi. Pembuatan interface vxlan0 pada server PVE1 menggunakan template Jinja2 yang tersimpan pada file *templates.j2*, sedangkan pada server PVE2 menggunakan file *templatess.j2*. Terakhir, *Playbook Ansible* ini juga memuat tugas untuk me-restart layanan jaringan agar perubahan konfigurasi jaringan virtual pada *file /etc/network/interfaces* dapat diaktifkan. Cuplikan isi file *playbook Create-Virtual-Network.yml* dengan tugas-tugas yang dijalankan pada server PVE1 ditunjukkan pada Gambar 7.”.



Gambar 7. File Playbook Ansible.

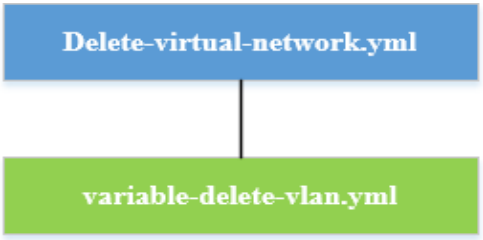
Struktur playbook Ansible untuk menambahkan jaringan virtual ditunjukkan pada Gambar 8.



Gambar 8. Struktur playbook Ansible.

File playbook utama untuk proses penambahan atau pembaruan jaringan virtual disebut *Update-virtual-network.yml*. File ini memuat tugas-tugas, termasuk penambahan 10 (sepuluh) VLAN baru dengan ID 21 hingga 30, beserta pengaturan alamat IP pada interface yang datanya diambil dari file *update-VLAN-pve1.yml* untuk diterapkan pada server PVE1, dan *update-VLAN-pve2.yml* untuk diterapkan pada server PVE2. Selain itu, playbook ini juga memuat tugas untuk me-restart layanan jaringan agar perubahan konfigurasi jaringan virtual pada *file /etc/network/interfaces* dapat diaktifkan. Sementara itu, struktur playbook Ansible untuk

penghapusan jaringan virtual ditunjukkan pada Gambar 9.

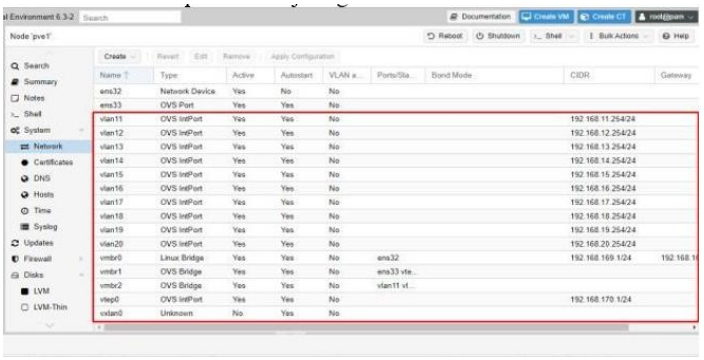


Gambar 9. Struktur Playbook Ansible dalam Pembuatan Jaringan Virtual.

File playbook utama untuk proses penghapusan jaringan virtual *Delete-virtual-network.yml.* File ini memuat tugas-tugas yang mencakup seluruh VLAN pada setiap server PVE, dengan data yang diambil dari file *variable-delete-VLAN.yml.* Selain itu, playbook ini juga memuat tugas untuk menghapus interface jembatan vmbr1, vmbr2, dan vxlan0. Layanan jaringan kemudian di-restart agar perubahan konfigurasi jaringan virtual pada file */etc/network/interfaces* dapat diaktifkan.

Pengujian Sistem Konfigurasi Jaringan Virtual Secara Manual dan Otomatis

Pengujian sistem konfigurasi jaringan virtual dilakukan baik secara manual melalui antarmuka web (GUI) masing-masing server PVE maupun secara otomatis sebanyak 5 (lima) kali percobaan. Pengujian otomatis dilakukan dengan menjalankan perintah ansible-playbook Create-Virtual-Network.yml pada server CentOS 7, yang berfungsi sebagai mesin kontrol Ansible. Cuplikan hasil verifikasi pembuatan jaringan virtual secara otomatis melalui GUI web dari server PVE1 ditunjukkan pada Gambar 10.



Gambar 10. Verifikasi Otomatis Hasil Pembuatan Jaringan Virtual pada Server PVE1.

Terlihat bahwa interface jaringan virtual, termasuk vmbr1, vmbr2, vtep0, vxlan0, serta sepuluh VLAN dengan ID 11 hingga 20, telah berhasil dibuat, termasuk pengaturan alamat IP pada masing-masing interface tersebut. Sementara itu, operasi penambahan jaringan virtual dilakukan dengan menjalankan perintah *ansible-playbook Update-virtual-network.yml.* Sedangkan operasi penghapusan jaringan virtual dilakukan dengan menjalankan perintah *ansible-playbook Delete-virtual-network.yml* melalui terminal pada server *CentOS 7.*

Analisis Sistem Otomatisasi Konfigurasi Jaringan Virtual

Analisis berikut diperoleh berdasarkan hasil pengujian sistem otomatisasi konfigurasi jaringan virtual berbasis OVS yang telah dilakukan pada 2 (dua) server PVE. Playbook Ansible yang dibuat dapat digunakan untuk mengotomatisasi proses pembuatan jaringan virtual berbasis OVS pada setiap server PVE, menambahkan sepuluh VLAN baru, serta melakukan penghapusan jaringan virtual. Uji verifikasi koneksi antar container pada VLAN yang sama tetapi berada di server PVE berbeda berhasil dilakukan menggunakan utilitas ping, yang menunjukkan bahwa VXLAN berfungsi dengan baik. Perbandingan waktu pembuatan jaringan virtual berbasis OVS pada masing-masing server PVE dilakukan melalui 5 (lima) percobaan secara manual maupun otomatis, sebagaimana ditunjukkan pada Tabel 3.

Tabel 3. Analisis Perbandingan Waktu Untuk *Create*Jaringan Virtual Manual dan Otomatis di Server PVE1 & PVE2.

Trial	Manual	Otomatis
First	11 Menit 15 Detik	1 Menit 24 Detik
Second	10 Menit 25 Detik	2 Menit 15 Detik
Third	10 Menit 09 Detik	2 Menit 15 Detik
Fourth	09 Menit 01 Detik	2 Menit 30 Detik
Fifth	07 Menit 09 Detik	2 Menit 30 Detik
Average	10 Menit 11 Detik	2 Menit 30 Detik
Minimum (Fastest Time)	09 Menit 01 Detik	01 Menit 24 Detik
Maximum (Longest Time)	11 Menit 15 Detik	02 Menit 30 Detik

Tabel 3 memperlihatkan bahwa rata-rata waktu untuk menambahkan jaringan virtual secara manual adalah 10 menit 4 detik, sedangkan dengan otomatisasi hanya membutuhkan 1 menit 47 detik. Artinya, penambahan jaringan virtual secara otomatis lebih cepat sekitar 8 menit 17 detik dibandingkan metode manual. Waktu tercepat untuk penambahan manual tercatat 8 menit 44 detik, sedangkan waktu terlama mencapai 11 menit. Sementara itu, penambahan otomatis memiliki waktu tercepat 1 menit 40 detik dan waktu terlama 2 menit 1 detik. Selanjutnya, Tabel 4 menyajikan perbandingan durasi proses penghapusan jaringan virtual berbasis OVS pada masing-masing server PVE. Proses ini diuji sebanyak lima kali percobaan, baik secara manual maupun melalui Ansible playbook.

Tabel 4. Perbandingan Waktu untuk *Update* VLAN pada Server PVE1 dan PVE2 02 menit 01 second.

Trial	Manual	Automatic
Fifth	09 Menit 51 Detik	01 Menit 40 Detik
Average	10 Menit 04 Detik	01 Menit 47 Detik
Minimum (Fastest Time)	08 Menit 44 Detik	01 Menit 40 Detik
Maximum (Longest Time)	11 Menit 00 Detik	01 Menit 47 Detik

Tabel 4 menunjukkan bahwa rata-rata waktu untuk membuat jaringan virtual secara manual adalah 11 menit 8 detik, sedangkan dengan otomatisasi hanya membutuhkan 2 menit 25 detik. Artinya, pembuatan jaringan virtual secara otomatis lebih cepat 8 menit 42 detik dibandingkan cara manual. Selain itu, waktu tercepat untuk pembuatan manual adalah 10 menit 6 detik, sedangkan yang terlama mencapai 12 menit 18 detik. Sementara itu, secara otomatis, waktu tercepat adalah 2 menit 20 detik dan terlama 2 menit 36 detik. Perbandingan waktu ini dilakukan pada proses penambahan jaringan virtual berbasis OVS pada setiap server PVE sebanyak 10 VLAN (dengan ID 21 hingga 30), yang diuji dalam 5 percobaan menggunakan metode manual maupun otomatis melalui playbook Ansible, sebagaimana ditunjukkan pada Tabel 5.

Tabel 5. Perbandingan Waktu untuk delete VLAN pada Server PVE1 & Server PVE2.

Trial	Manual	Automatic
First	03 Menit 48 Detik	02 Menit 37 Detik
Second	03 Menit 03 Detik	02 Menit 41 Detik
Third	03 Menit 39 Detik	02 Menit 36 Detik
Fourth	03 Menit 10 Detik	02 Menit 38 Detik
Fifth	03 Menit 05 Detik	02 Menit 43 Detik
Average	03 Menit 21 Detik	02 Menit 39 Detik
Minimum (Fastest Time)	03 Menit 03 Detik	02 Menit 36 Detik
Maximum (Longest Time)	03 Menit 48 Detik	02 Menit 43 Detik

4. KESIMPULAN

Berdasarkan analisis dan pengujian yang dilakukan, dapat disimpulkan bahwa sistem otomatisasi berbasis Ansible terbukti efektif dalam pengelolaan konfigurasi jaringan virtual Open vSwitch pada dua server Proxmox VE. Sistem ini mampu melaksanakan seluruh proses, termasuk pembuatan, penambahan, dan penghapusan bridge, VLAN, serta VXLAN, dengan tingkat keberhasilan yang tinggi. Hasil evaluasi menunjukkan bahwa otomatisasi secara signifikan mempercepat manajemen jaringan virtual dibandingkan metode manual, dengan rata-rata waktu percepatan masing-masing 7 menit 43 detik untuk pembuatan, 7 menit 16 detik untuk penambahan, dan 39 detik untuk penghapusan. Selain peningkatan efisiensi waktu, penerapan otomatisasi juga menurunkan kemungkinan kesalahan manusia, memastikan konsistensi konfigurasi, serta mendukung pengelolaan jaringan yang lebih andal dan skalabel. Dengan demikian, pemanfaatan Ansible untuk otomatisasi jaringan virtual menawarkan solusi yang praktis dan efektif dalam manajemen infrastruktur cloud berbasis Proxmox VE.

DAFTAR REFERENSI

Anjani, D. N., & Wardhani, M. (2024). Family farming: Pembuatan pestisida nabati dari daun sirsak untuk pengendalian hama pada tanaman sayuran di Desa Belo Kecamatan Palibelo Kabupaten Bima. *5*(2), 127–134. https://doi.org/10.36312/abdimandalika.v5i2.4112

Damanik, H. A., & Anggraeni, M. (2025). Manajemen jaringan terpusat untuk konfigurasi dan otomatisasi pemulihan menggunakan Paramiko dan Django Framework/Centralized network management for configuration and recovery automation using Paramiko and Django Framework. *11*, 369–382. https://doi.org/10.28932/jutisi.v11i3.11431

Emmungil, L. (2025). An empirical analysis of server utilization and optimization strategies in a Proxmox VE environment. *2*(2).

Grantor, D. (2026). *Sveučilišni diplomski studij računarstva: Diplomski rad napredno upravljanje konfiguracijom virtualnih testnih mreža korištenjem Nixa.*

Intelligence, D., Technology, I., Universitas Pembangunan Panca Budi, & Pudi. (2025). Proxmox-based virtualization techniques on virtual servers, 351–357.

Izoulet, D., Beserra, D., Espie, M., Endo, P. T., & Araujo, J. (n.d.). *Performance evaluation of Proxmox for high-performance computing in resource-shared environments.*

Koczubiej-Nowakowska-Stąpór-Świetlik-Walczyk. (n.d.). *204-Koczubiej-Nowakowska-Stąpór-Świetlik-Walczyk.pdf*.

Mahendra, K. G. S., Suardika, I. G., & Santosa, I. M. A. (2025). Implementasi sistem monitoring dan logging menggunakan Ansible pada project di PT Itsavirus. *2*(1), 967–972.

Minahasa. (2024). *3 1,2,3. 24*(7), 28–42.

Oleksiuk, V. P., & Oleksiuk, O. R. (n.d.). *The practice of developing the academic cloud using the Proxmox VE platform.* 0272.

Pitra, M. D., et al. (2025). Implementasi load balancer HAProxy dengan konfigurasi dinamis dan autoscaling Nginx web. *9*(7), 1–8.

Pramadani, A., et al. (2025). Implementasi sistem dynamic link offloading berbasis Ansible pada router konvensional dengan jaringan. *9*(9), 1–8.

Putra, D. (2025). Implementation of server provisioning and hardening automation using Terraform and Ansible with NIST SP. *2*, 1354–1364. https://doi.org/10.62201/9krgf663

Syaifuddin, F. A., & Amrulloh, M. F. (2025). Analisis efisiensi Ansible dalam manajemen patch dan update pada Linux dengan pendekatan DevOps. *Digitech, 5*(2), 20–30. https://doi.org/10.47709/digitech.v5i2.6681

Vitor. (2025). *Architecture for scalable deployment of AI models.*

Wardhani, & Nuruzzaman, M. T. (2023). Analisis pengaruh penggunaan Internet Download Manager pada load balancing di Mikrotik. *Edumatic: Jurnal Pendidikan Informatika, 7*(2), 495–504. https://doi.org/10.29408/edumatic.v7i2.24295

Widjajarto, A., Yunan, U., & Septo, K. (2025). Implementasi dan analisa security hardening pada server Linux menggunakan Ansible. *3*(2), 94–101. https://doi.org/10.25124/jpeia.v3i2.9911

Zulfikar, A., & Akbar, Y. (2025). Automation of Mikrotik router setting configuration backup using Ansible with Network DevOps method/Otomasi backup konfigurasi settingan router Mikrotik menggunakan Ansible dengan metode Network DevOps. *5*(1), 57–66. https://doi.org/10.57152/malcom.v5i1.1591