Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.202293947
Vol. 9, No. 1, Februari 2022, hlm. 79-86 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

NETWORK AUTOMATION PADA BEBERAPA PERANGKAT *ROUTER* MENGGUNAKAN PEMROGRAMAN PYTHON

Setiya Nugroho[*1], Bambang Pujiarto[2]

[1,2] Universitas Muhammadiyah Magelang, Magelang
Email: [1]setiya@ummgl.ac.id, [2]amadheos@gmail.com
[*]Penulis Korespondensi

(Naskah masuk: 22 Agustus 2020, diterima untuk diterbitkan: 27 Januari 2022)

Abstrak

Perkembangan teknologi dalam bidang jaringan komputer memberikan efek semakin banyak vendor yang memproduksi perangkat jaringan. Perancangan *topologi* jaringan dengan tipe *Wide Area Network* (WAN) membutuhkan lebih banyak perangkat jaringan termasuk *router*. Seorang *network administrator* (*Netadmin*) yang melakukan konfigurasi lebih dari satu *router* akan memberi peluang adanya *human error*. Selain itu, *Netadmin* membutuhkan waktu semakin banyak untuk menyelesaikan konfigurasi karena harus berpindah dari satu *interface* ke *interface* lain. Untuk mengatasi permasalahan tersebut, pada penelitian ini telah dirancang sebuah aplikasi *network automation* untuk mengatur beberapa perangkat *router*. Tujuan penelitian ini adalah membuat sebuah *dashboard* berbasis web yang dapat mengontrol beberapa *router* melalui satu *interface*. Metode yang digunakan dalam penelitian ini melalui beberapa tahap. Tahap pertama adalah perancangan *topologi* perangkat keras pada *network automation*. Perancangan topologi menggunakan beberapa perangkat yaitu enam buah *router*, satu buah *switch hub*, dan satu buah computer *host*. Tahapan kedua adalah perancangan perangkat lunak menggunakan *Unified Modelling Language* (UML). Pemodelan UML pada riset ini memanfaatkan *use case diagram* dan *activity diagram*. Pengujian yang digunakan pada penelitian ini adalah metode *white box* dan *black box testing*. Hasil penelitian ini telah dibangun sebuah aplikasi *network automation* berbasis web menggunakan pemrograman Python dengan *framework* Django dan *library paramiko*. Aplikasi telah diuji coba untuk melakukan konfigurasi tiga buah *router* Cisco dan tiga buah *router* Mikrotik secara bersamaan.

Kata kunci: *network automation, Python, Django, paramiko*

NETWORK AUTOMATION IN SOME ROUTER DEVICES USING PYTHON PROGRAMMING

Abstract

Technological developments in the field of computer networks affect the increasing number of vendors producing network devices. Designing a network topology with a Wide Area Network (WAN) type requires more network devices including routers. A network administrator (Netadmin) configuring more than one router will provide an opportunity for human error. Besides that, Netadmin takes more time to complete configuration because it has to move from one interface to another. To solve this problem, this research has designed a network automation application to manage multiple router devices. The purpose of this research is to create a web-based dashboard that can control multiple routers through a single interface. The method in this study using several stages. The first stage is designing a hardware topology for network automation. The topology design uses several devices, namely six routers, one switch hub, and one host computer. The second stage is software design using the Unified Modeling Language (UML). UML modeling in this research utilizes use case *diagrams and activity diagrams. The test used in this study is white box and black box testing methods. The results of this study have built a web-based network automation application using Python programming with the Django framework and the Paramiko library. The application has been tested to configure three Cisco routers and three Mikrotik routers simultaneously.*

Keywords: *network automation, Python, Django, paramiko*

1. PENDAHULUAN

Perkembangan teknologi informasi dan komunikasi (TIK) saat ini mencakup segala aspek kehidupan. Penerapan teknologi yang memudahkan setiap orang untuk mengakses dan berbagi informasi menjadi penyebab perkembangan TIK yang pesat. Salah satu penerapan TIK yang berkembang cepat

adalah jaringan komputer. Saat ini hampir semua perusahaan menggunakan jaringan komputer untuk berkomunikasi dengan pihak lain dengan aturan yang saling disepakati (Rahmawati, 2015).

Sejalan dengan perkembangan TIK, perusahaan memerlukan peralatan pendukung jaringan komputer. Peralatan tersebut kini menjadi komponen penting dalam pembangunan jaringan komputer. *Router* sebagai salah satu komponen pada jaringan komputer mampu melewatkan data melalui sebuah jaringan internet atau intranet menuju sasarannya, melalui sebuah proses *routing*. *Router* berfungsi sebagai penghubung antar dua atau lebih jaringan untuk meneruskan data dari satu jaringan ke jaringan lainnya (Asnawi, 2018; Handoyo, 2011).

Jaringan komputer saat ini semakin kompleks dan dinamis. Jaringan komputer yang kompleks akan membutuhkan banyak perangkat jaringan termasuk *router* dengan berbagai jenis, tipe dan merk. Jaringan kompleks saat ini biasanya melibatkan integrasi dan interkoneksi banyak perangkat *router*. Operator jaringan bertanggung jawab untuk mengonfigurasi jaringan dan merespons peristiwa yang terjadi di jaringan. Operator jaringan harus menerapkan tugas kompleks dengan serangkaian perintah konfigurasi dalam lingkungan *command line interface* (CLI). Konfigurasi pada jaringan akan semakin sulit implementasinya karena sebuah kebijakan pada level pimpinan akan berakibat penerapan konfigurasi yang berbeda pada beberapa *router* dengan jenis, tipe dan merk yang berbeda. Kondisi jaringan dinamis berarti kondisi jaringan yang terus berubah. Penambahan perangkat, peningkatan jumlah pemakai adalah beberapa penyebab jaringan bersifat dinamis. Operator jaringan harus menyesuaikan konfigurasi jaringan sebagai tanggapan terhadap perubahan kondisi jaringan (Balaram et al., 2016).

Jaringan komputer yang kompleks dan dinamis mengakibatkan operator jaringan akan semakin sulit melakukan konfigurasi dan pengaturan. Operator jaringan mengetikkan secara manual perintah konfigurasi menggunakan CLI pada sekian banyak *router* dengan perbedaan jenis, tipe dan merk dan setiap ada perubahan kebijakan. Operator jaringan harus melakukan login setiap kali berganti *router*, sehingga semakin banyak *router* semakin banyak waktu dibutuhkan untuk melakukan konfigurasi. Akibat perubahan konfigurasi sering terjadi, operator harus selalu membutuhkan konsentrasi yang tinggi dalam setiap melalukukan konfigurasi. Kondisi demikian akan memungkinkan operator sering melakukan kesalahan konfigurasi pada setiap *router*.

Operator jaringan memerlukan cara yang lebih baik untuk mengkonfigurasi dan mengelola jaringan. Sofware Defined Networking (SDN) bisa menjadi salah satu inovasi teknologi dalam manajemen jaringan sekarang ini. Open Networking Foundation (ONF) mengembangkan OpenFlow sebagai *protocol* untuk mengadopsi konsep SDN (Braun & Menth, 2014). Namun, tidak semua perangkat *router* support dengan *protocol* OpenFlow. Misalnya untuk merk Mikrotik, mulai versi 6.rc8 yang sudah menggunakan OpenFlow. Sehingga, perangkat *router* dengan versi lama yang diproduksi oleh beberapa vendor tidak bisa diterapkan teknologi SDN karena belum mendukung *protocol* OpenFlow. Oleh karena itu, perlu adanya sebuah teknologi yang mendukung berbagai perangkat *router* konvensional sekaligus perangkat modern untuk mengelola jaringan yang lebih mudah. Pada penelitian ini, telah dibuat sebuah *network automation* pada beberapa perangkat *router* baik konvensional maupun modern.

Ratan (2018) menginformasikan bahwa seorang *automation engineer* bisa menggunakan PowerShell atau Python sebagai bahasa bahasa pemrograman untuk *network automation*. Microsoft memberikan dukungan penuh kepada PowerShell. Secara khusus, Microsoft akan memberikan update penting kepada PowerShell 5.0 pada sistem operasinya mulai dari Windows 10. Sementara Python mendapat kontribusi dari ribuan developer karena sifatnya open source. Python memiliki sub program Paramiko dan Netmiko yang digunakan mendefinisikan network *router*s dari vendor tertentu. Pyhton mendukung kolaborasi aman dari beberapa pengguna dengan transaksi data yang terenkripsi menggunakan GitHub (Ratan, 2018).

Risei ini memilih menggunakan Python dengan beberapa alasan. Python adalah bahasa pemrograman tingkat tinggi yang mudah dipelajari, menjadi bagian integral dari jaringan skala besar, dan memecahkan masalah untuk mempersingkat operasi pada jaringan (Chou, 2018). Python menjadi andalan bagi network administrator untuk memonitor, *konfigurasi* dan administrasi jaringan. Network admnistrator dapat melakukan monitor secara *real-time* apa yang sedang terjadi di jaringan menggunakan Python. Selain itu, netadmin tidak perlu mengetikkan perintah dengan CLI untuk mengkonfigurasi jaringan yang berulang karena dengan pyhon bisa membuat antarmuka secara *graphical* (Chou, 2020). Python adalah bahasa pemrograman yang memiliki fitur lengkap dengan library yang terdokumentasi dengan baik, sehingga akan memudahkan network programming untuk bereksperimen membuat program. Seorang network programming akan dengan mudah mengambil dan meminta data dari sebua web, serta dengan mudah melakukan ekstraksi data menjadi format yang umum melalui web hanya menggunakan Python. Network programming dapat memanfaatkan Python untuk membuat program email, menggunakan berbagai *internet protocol*, dan melakukan sistem *remote* dan DNS *networking* (Ortega et al., 2019)

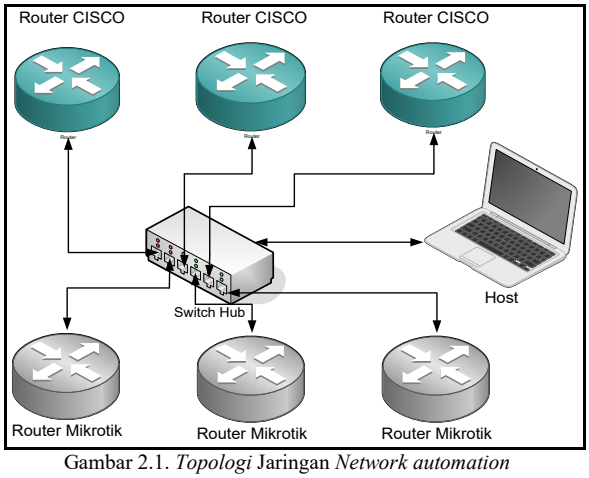
Ada beberapa penelitian yang berkaitan dengan *network automation*. Mihăilă, dkk (2017) melakukan penelitian *network automation and abstraction*. Mihăilă membuat *network automation* menggunakan *library* Python : *netmiko* dan *paramiko* pada Ubuntu Docker Container di lingkungan GNS3 emulator (Mihăilă et al., 2017). Swastika (2017) melakukan uji coba *network automation* pada *router* Mikrotik

menggunakan software Ansible dengan konfigurasi pengaturan bandwidth memafaatkan fitur dari *Queue Tree* (Swastika & Atitama, 2017). Wijaya (2019) meneliti *network automation* menggunakan software Ansible yang diinstall pada Ubuntu Desktop untuk mengelola perangkat *router* Cisco pada emulator Eve-ng (Wijaya, 2019). Komarudin (2018) membuat riset tentang *network automation* menggunakan beberapa *library* Python : *netmiko*, *paramiko*, *pyntc*, *napalm* dan tools Ansible. Komarudin mengimplementasikan semua tools tersebut pada linux Ubuntu versi 16.04 untuk mengkonfigurasi *router* Cisco (Komarudin, 2018).

Pada eksperimen kali ini telah dibuat network automation menggunakan pemrograman Python dengan *library paramiko* untuk mengkonfigurasi beberapa perangkat *router*. Swastika (2017) hanya menggunakan *router* Mikrotik saja, Wijaya (2019) hanya memanfaatkan *router* Cisco saja, sedangkan riset ini memakai kedua *router* tersebut. Studi tidak memanfaatkan sistem operasi linux Ubuntu seperti peneliti sebelumnya. Mihăilă (2017) menggunakan Ubuntu Docker Container, Wijaya (2019) memakai Ubuntu Desktop, sedangkan Swastika (2017) dan Komarudin (2018) menerapkan Ubuntu versi 16.04. Observasi ini mengembangkan aplikasi berbasis web dengan memanfaatkan *framework* Django sehingga tidak harus menerapkan sistem operasi tertentu termasuk Windows 10 yang digunakan di riset ini.

2. METODE PENELITIAN

Tahapan pertama perancangan pada penelitian ini adalah pembuatan *topologi* perangkat keras. Pada riset kali ini, telah digunakan tiga buah *Router* CISCO, tiga buah *Router* Mikrotik, satu switch hub dan sebuah laptop seperti terlihat pada gambar 2.1.



Gambar 2.1. *Topologi* Jaringan *Network automation*

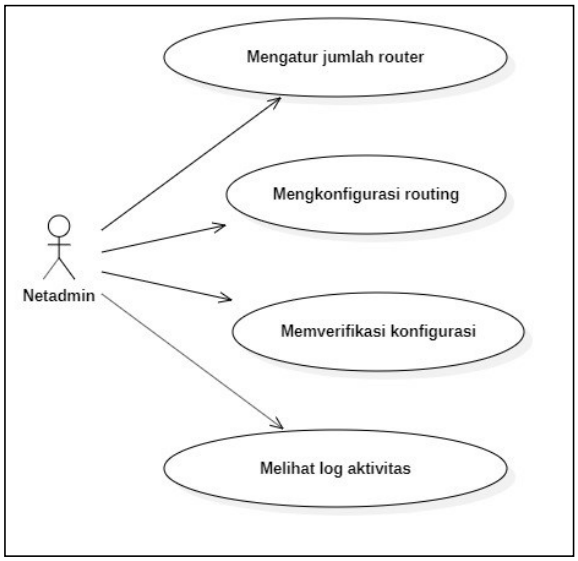
Riset ini menggunakan tiga buah *Router* CISCO yaitu seri C2600, C3660 dan C7200, tiga buah *Router* Mikrotik yaitu seri chr-6.45, chr-6.46 dan chr-6.47. Eksplorasi menerapkan laptop sebagai host yang akan dijadikan *web server* untuk pengembangan aplikasi sebagai tampilan antarmuka (*interface*). Switch hub diperlukan sebagai perantara untuk menyambungkan tiga *Router* CISCO, tiga *Router* Mikrotik dan host.

Tahapan kedua perancangan pada penelitian ini adalah pembuatan desain untuk tampilan antarmuka dari perangkat lunak yang akan dibangun. Eksplorasi ini menerapkan *Unified Modelling Language* (UML) sebagai perancangan aplikasi web berbasis *Object Oriented Programming* (OOP).

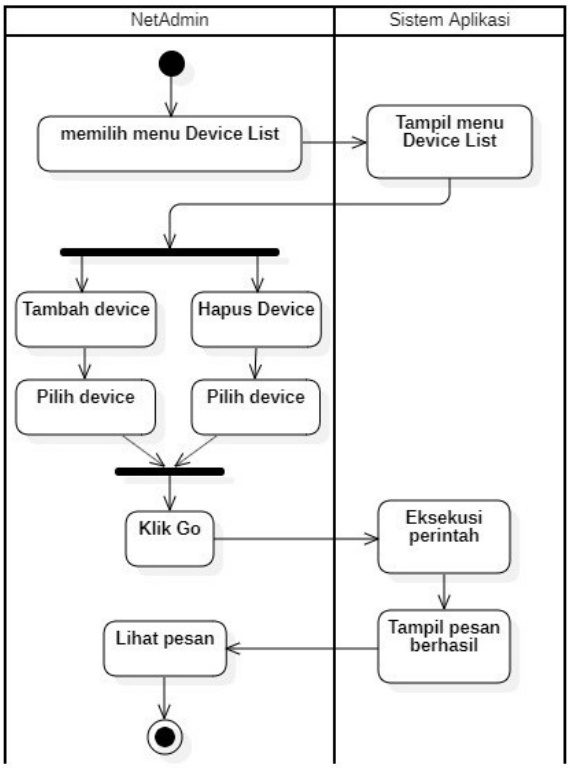
Menurut Lott (2019) dalam bukunya berjudul Mastering Object-Oriented Python Second Edition, Python adalah salah satu bahasa pemrograman OOP atau berorientasi objek yang mempermudah dalam mengeksplorasi penggunaan algoritma dan data struktur yang berbeda yang menghasilkan alternatif performa komputer. (Lott, 2019)

Tahapan awal pada perancangan UML adalah pembuatan *use case diagram*. Pada *use case diagram*, telah diidentifikasikan Netadmin sebagai aktor atau pengguna. Netdamin memiliki beberapa fungsi pada aplikasi antarmuka *network automation*. Aplikasi ini akan diberi nama Setnema, yang merupakan akronim dari Setiya *network automation* Netadmin pada aplikasi Setnema mempunyai fungsi mengatur jumlah *router*, melakukan konfigurasi *routing*, memverifikasi konfigurasi *routing*, dan melihat log aktivitas jaringan. *Use case diagram* pada aplikasi Setnema terlihat pada gambar 2.2.

Tahapan berikutnya pada perancangan UML ini adalah pembuatan *activity diagram*. *Activity diagram* dibuat berdasarkan jumlah fungsi yang ada pada *use case diagram*. Sehingga jumlah *activity diagram* pada aplikasi Setnema ini adalah empat. *Activity diagram* yang pertama adalah mengatur jumlah *router* seperti yang terlihat pada gambar 2.3.



Gambar 2.2. *Use Case Diagram* Aplikasi Setnema

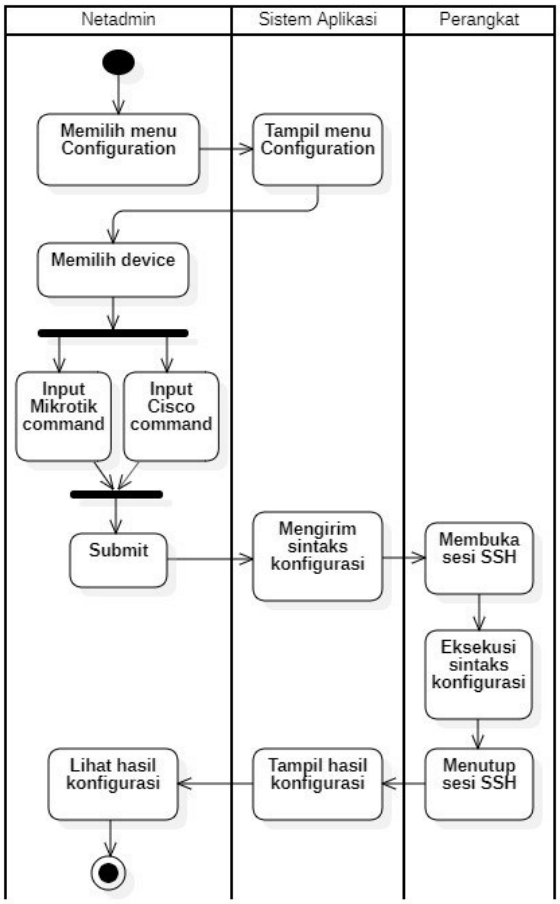


Gambar 2.3. *Activity Diagram* mengatur jumlah *router* aplikasi Setnema

Pada *activity diagram* mengatur jumlah *router*, Netadmin bisa menambah atau menghapus jumlah *router* yang akan digunakan pada aplikasi Setnema. Setelah penambahan atau penghapusan jumlah *router*, Netadmin bisa melihat pesan yang ditampilkan. Netadmin hanya berinteraksi dengan sistem aplikasi saja pada *activity diagram* mengatur jumlah *router*.

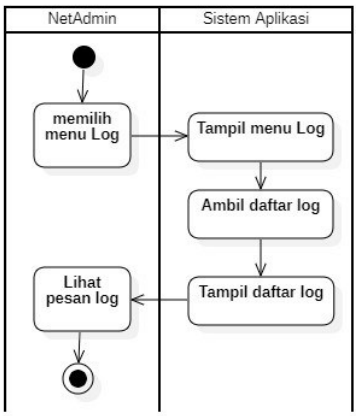
Activity diagram kedua pada aplikasi Setnema adalah mengkonfigurasi *routing*. Neadmin memiliki aktivitas memilih *device router* yang telah ditentukan pada aktivitas sebelumnya. Berikutnya, Netadmin bisa menginputkan perintah pada *router* Mikrotik dan Cisco sekaligus. Sistem aplikasi akan meneruskan perintah sintaks konfigurasi dari Netadmin ke *device* perangkat *router* dengan membuka sesi SSH terlebih dahulu. Perngakat mengeksekusi sintaks konfigurasi yang diawali membuka sesi SSH dan diakhir menutup sesi SSH. Pada bagian akhir, sistem aplikasi akan menampilkan hasil konfigurasi, sehingga Netadmin bisa melihat hasil konfigurasi. *Activity diagram* mengkonfigurasi *routing* terlihat pada gambar 2.4

Activity diagram memverifkasi konfigurasi adalah *activity diagram* berikutnya. *Activity diagram* ini mirip dengan *activity diagram* mengkonfigurasi *routing*. Pada *activity diagram* ini, Netadmin juga mempunyai aktivitas memilih *device* terlebih dahulu. Setelah itu, Netadmin akan memasukkan perintah pada *device* Mikrotik dan Cisco cukup sekali untuk semua *router* dengan merk yang sama. Perbedaan dengan *activity diagram* mengkonfigurasi *routing* terletak pada laporan hasil yang ditampilkan.



Gambar 2.4. *Activity Diagram* mengkonfigurasi *routing* aplikasi Setnema

Pada *activity diagram* mengkonfigurasi *routing*, Netadmin menginputkan sintaks konfigurasi *routing* dan hasilnya hanya ditampilkan hasil konfigurasi *routing* berhasil atau gagal. Sedangkan pada *activity diagram* memverifkasi konfigurasi, Netadmin menginputkan sintaks konfigurasi *routing* dan hasil dari tampilan console juga akan ditampilkan pada *activity diagram* memverifkasi konfigurasi.



Gambar 2.4. *Activity Diagram* melihat log aplikasi Setnema

Activity diagram terakhir adalah melihat log aktivitas seperti terlihat pada gambar 2.5. Pada

activity diagram ini, Netadmin bisa melihat aktivitas yang berupa terget *device*, jenis aksi, status dan waktu tanggal.

3. HASIL DAN PEMBAHASAN

3.1. Hasil

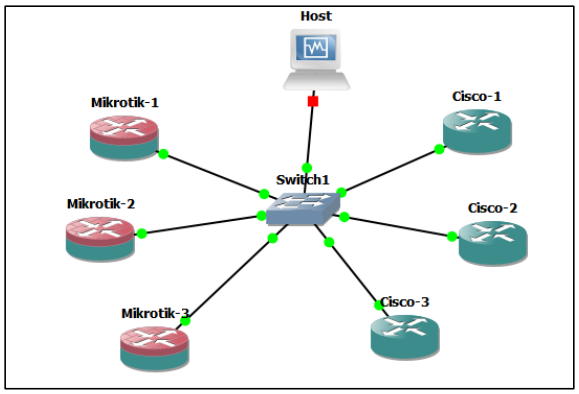
Pada observasi ini digunakan aplikasi GNS3 versi 2.2.8 sebagai simulator jaringan pada *network automation*. *Router* yang digunakan sesuai dengan perancangan perangkat yaitu tiga buah *Router* Cisco dan tiga buah *Router* Mikrotik. Setiap *router* akan dialokasikan sejumlah memori sesuai rekomendasi dokumentasi pada GNS3. Alokasi memori dan *IP Address* pada setiap *device router* terlihat pada tabel 3.1, sedangkan implementasi *topologi* jaringan *Network automation* menggunakan GNS3 terlihat pada gambar 3.1.

Tabel 3.1. Alokasi memori dan *IP Address* pada *device router*

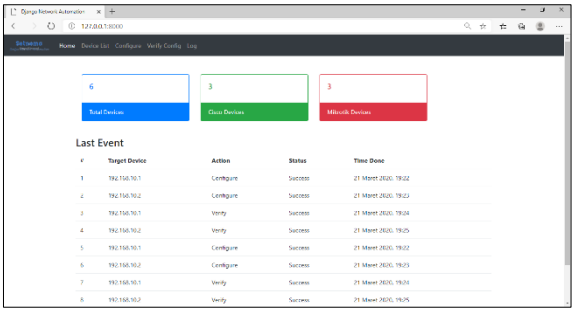
No	*Router*	Seri	Memori	*IP Address*
1	Cisco-1	C2600	128 MB	169.254.99.3
2	Cisco-2	C3660	192 MB	169.254.99.4
3	Cisco-3	C7200	512 MB	169.254.99.5
4	Mikrotik-1	chr-6.45	256 MB	169.254.99.6
5	Mikrotik-2	chr-6.46	256 MB	169.254.99.7
6	Mikrotik-3	chr-6.47	256 MB	169.254.99.8

Ada beberapa konfigurasi yang ditambahkan pada perangkat keras. Riset ini perlu ditambahkan konfigurasi SSH pada setiap *router* agar *router* bisa diremote oleh sistem aplikasi Setnema. Pada komputer host, telah diinstall Python versi 3.7.2 sebagai bahasa pemrograman untuk membuat backend pada aplikasi Setnema. Paramiko versi 2.7.1 ditambahkan sebagai library Python. Paramiko bertindak menjadi SSH client yang akan meremote perangkat jaringan yaitu *Router* Cisco dan Mikrotik.

Aplikasi Setnema dibangun menggunakan web *framework* Django 3.0.8. Django dibuat dari bahasa pemrograman Python, dengan *source code* yang jelas dan mudah dibaca. Django dirancang khusus untuk membantu pengembang membangun aplikasi web cepat dan efisien. (Kronika & Bendoraitis, 2018)



Gambar 3.1. Implementasi *topologi* jaringan *network automation*



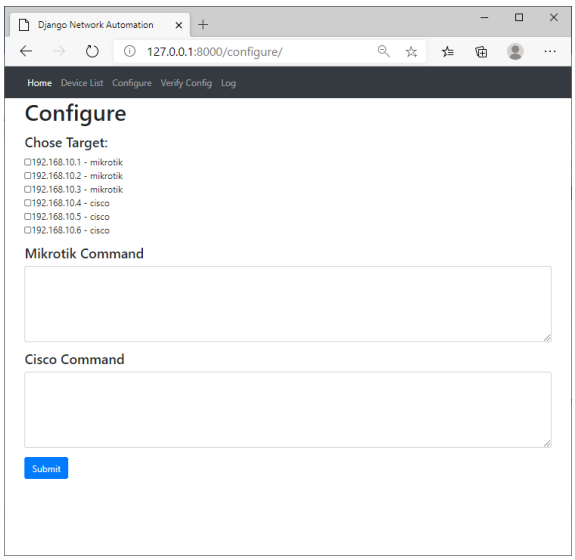
Gambar 3.2. Halaman *dashboard* aplikasi Setnema

Tampilan frontend aplikasi Setnema dibangun menggunakan CSS *framework* Bootstrap. Bootstrap mempermudah membuat tampilan aplikasi Setnema menjadi responsive dengan menggabungkan teknologi HTML, CSS dan Javacript.

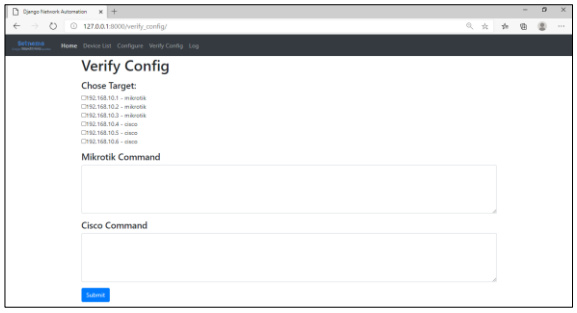
Gambar 3.2. merupakan tampilan dari halaman *dashboard* aplikasi Setnema. Halaman *dashboard* menampilkan jumlah total *device* perangkat *router*, jumlah *Router* Cisco dan jumlah *router* Mikrotik yang telah ditambahkan oleh Netadmin. Selain itu, halaman *dashboard* juga menampilkan sepuluh aktivitas terakhir yang dilakukan oleh Netadmin.

Menu *Device List* akan menampilkan semua perangkat jaringan berupa *Router* Mikrotik dan Cisco yang telah ditambahkan pada bagian beckend. *Device List* menampilkan perangkat jaringan dengan rincian *IP Address*, Hostname dan Vendor tiap perangkat.

Menu *Configure* akan memberikan pilihan kepada Netadmin sejumlah *router* yang bisa dipilih dan akan dikonfigurasi. Isian *text box* hanya ada dua yang mewakili *Router* Mikrotik dan Cisco. Netadmin yang mengisi konfigurasi pada *text box* Mikrotik maka otomatis akan dikirim ke semua perangkat yang dipilih Netadmin dengan merk Mikrotik. Hal yang sama juga diberlakukan pada *router* Cisco. Hasil dari inputan konfigurasi Netadmin pada menu *Configure* akan ditampilkan pada menu Log dengan hasil *error* atau *success*. Menu *Configure* terlihat gambar 3.3.



Gambar 3.3. Halaman menu *configure* aplikasi Setnema



Gambar 3.4. Halaman menu *Verify config* aplikasi Setnema

Menu *Verify config* memiliki tampilan yang mirip deengan menu *Configure* yaitu pilihan *router* yang akan dikonfigurasi, *text box* untuk input sintaks *routing* pada *router* Mikrotik dan Cisco. Jika hasil dari menu *Configure* hanya ditampilkan pada menu Log maka untuk menu *Verify config* akan menampilkan semua hasil seperti yang ditampilkan pada console masing-masing *router*.

Menu log akan menampilkan semua aktifitas dari Netadmin baik *Configure* maupun *Verify config*. Menu log akan memberi informasi berupa target *device* yaitu *IP Address* dari setiap *router*. Informasi log kedua adalah action yang berupa *configure* atau *verify config*. Informasi log berikutnya adalah status yaitu sukses atau *error* dan waktu eksekusi.

3.2. Pembahasan

Memori yang dialokasikan pada setiap *router* akan berbeda-beda. Untuk *router* Mikrotik minimal memori yang dialokasikan cukup 128 MB tetapi oleh rekomendasi dari dokumentasi GNS disarankan untuk menggunakan 256 MB memori. Sedangkan untuk *router* Cisco alokasi memori tergantung dari serinya. Semakin tinggi serinya maka semakin besar pula memori yang dialokasikan sesuai rekomendasi dari dokumentasi GNS3. Jumlah total memori yang harus dialokasikan sesuai dengan tabel 3.1 adalah 128 + 192 + 512 + 256 + 256 + 256 = 1600 MB.

Setiap *device* pada *topologi* jaringan *network automation* akan dipanggil dengan cara yang berbeda GNS3 bisa langsung memanggil image dari *router* Cisco dengan cara membuat *template* terlebih dahulu pada IOS *Router*s di Dynamips. GNS3 juga bisa memanggil image dari *router* Mikrotik dengan cara membuat *template* terlebih dahulu pada QEMU VMs. GNS3 memanggil Host OS dari vitual machine lain, pada percobaan kali ini digunakan Virtual Box. Host OS bisa juga disambungkan dengan parangkat fisik dengan cara membuat *loopback adapter* terlebih dahulu. Cara ini tidak banyak mengkonsumsi memori fisik, tetapi ada beberapa konfigurasi yang perlu dilakukan. Beberapa konfigurasi diantaranya yaitu membuat enable server localhost di bagian server pada menu *preference*, menginstall dan menjalankan *service* Winpcap atau Npcap, serta membuat *disable firewall* yang aktif. Tidak ada pengaruhnya bagi *network automation* dari kedua cara tersebut. Riset ini menggunakan Host OS pada Virtual Box.

```
def configure(request):
    if request.method == "POST":
        selected_device_id = request.POST.getlist('device')
        mikrotik_command = request.POST['mikrotik_command'].splitlines()
        cisco_command = request.POST['cisco_command'].splitlines()
        for x in selected_device_id:
            try :
                dev = get_object_or_404(Device, pk=x)
                ssh_client = paramiko.SSHClient()
                ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
                ssh_client.connect(hostname=dev.ip_address,username=dev.username,password=dev.password)

                if dev.vendor.lower() == 'cisco':
                    conn = ssh_client.invoke_shell()
                    conn.send("conf t\n")
                    for cmd in cisco_command:
                        conn.send(cmd + "\n")
                        time.sleep(1)
                else:
                    for cmd in mikrotik_command:
                        ssh_client.exec_command(cmd)
```

Gambar 3.5. Pengujian *white box* aplikasi Setnema

Pengujian aplikasi Setnema dilakukan dengan cara *white box* dan *black box* Untuk pengujian *white box*, riset ini menghilangkan script *import time* pada file *views.py*. Hasilnya ada peringatan pada menu Log yaitu *name time is not defined*. Hal ini terjadi karena pada bagian *configure* di file yang sama telah didefinisikan *time.sleep*. Selanjutnya, pada percobaan ini akan diterapkan *black box* testing dengan cara menginputkan perintah untuk menambahkan *interface loopback* paada kedua *router*. Perintah Mikrotik pada *text box* Mikrotik *command* dan perintah Cisco pada *text box* Cisco. Perintah tersebut perintah diinputkan pada menu *Configure*. Hasilnya muncul status *success* dan pesan *no error* pada menu Log. Untuk melihat pengaruh perintah tersebut pada *device router*, maka pada menu *Verify config* telah diinputkan perintah menampilkan semua *IP Address* pada pada *device router*. Hasilnya setiap *device router* akan menampilkan semua *IP Address* seperti tampilan pada console tiap *router*.

Aplikasi Setnema ini menggunakan *device router* Cisco dan Mikrotik saja Aplikasi *network automation* Setnema ini akan efektif bagi seorang Netadmin ketika pada jaringan dikelola terdapat banyak *router* dengan merk Cisco dan Mikrotik. Untuk perintah *routing* perlu dibuatkan menu baru sehingga seorang Netadmin bisa menginputkan dengan sekali aksi untuk semua *router* dengan merk yang sama.

4. KESIMPULAN DAN SARAN

Hasil dari penelitian ini dapat disimpulkan, aplikasi Setnema yang dibangun berhasil untuk melakukan *network automation* menggunakan bahasa pemrograman Python versi 3.7.2, *framework* Django versi 3.0.8, *libarary* Paramiko versi 2.7.1 dengan enam buah *router* di network simulator GNS3 versi 2.2.8. *Router* yang digunakan tiga *router* Cisco yaitu seri C2600, C3660, dan C7200. Tiga *router* Mikrotik yaitu seri chr-6.45, chr-6.46, dan chr-6.47. Total alokasi memori untuk keenam router adalah 1600 MB

Riset tidak berhasil jika digunakan *router* selain Cisco dan Mikrotik. Selain itu alokasi memori untuk router yang melebihi kapasitas memori fisik pada komputer host juga akan menghambat penelitian.

Saran untuk pengembangan selanjutnya aplikasi tersebut bisa digunakan untuk melakukan konfigurasi *routing* yang berbeda pada setiap *router*. Jenis *router* yang dikonfigurasi bisa ditambahkan Juniper, Aruba.

DAFTAR PUSTAKA

ASNAWI, M. F. 2018. Aplikas Konfigurasi Mikrotik Ssebagai Manajemen Bandwidth Dan Internet Gateway Berbasis Web. *Jurnal Penelitian Dan Pengabdian Kepada Masyarakat UNSIQ*, *5*(1), 42–48. https://doi.org/10.32699/ppkm.v5i1.437

BALARAM, V. V. S. S. S., MUKUNDHA, C., & BHUTADA, S. 2016. Enhancement of Network Administration through Software Defined Networks. *IOSR Journal of Computer Engineering*, *18*(1), 30–36. https://doi.org/10.9790/0661-18113036

BRAUN, W., & MENTH, M. 2014. Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices. *Future Internet*. https://doi.org/10.3390/fi6020302

CHOU, E. 2018. Mastering Python Networking Second Edition. In V. Boricha, P. Bisht, D. Thore, & V. Shivhare (Eds.), *Packt Publishing* (2nd ed.). Packt Publishing.

CHOU, E. 2020. Mastering Python Networking Third Edition. In *Packt Publishing* (3rd ed.). Packt Publishing.

HANDOYO, J. (2011). Kajian Penggunaan Mikrotik *Router* OSTM Sebagai *Router* Pada Jaringan Komputer. *Jurnal Transformatika*, *9*(1), 20. https://doi.org/10.26623/transformatika.v9i1.54

KOMARUDIN, A. R. 2018. *Otomatisasi Administrasi Jaringan Dengan Script Python*. Jasakom.

KRONIKA, J., & BENDORAITIS, A. 2018. Django 2 Web Development Cookbook ThirdEdition. In *Packt Publishing* (3rd ed.). Packt Publishing.

LOTT, S. F. 2019. Mastering Object-Oriented Python Second Edition. In *Packt Publishing* (2nd ed.). Packt Publishing. https://doi.org/10.1017/CBO9781107415324.004

MIHĂILĂ, P., BĂLAN, T., CURPEN, R., & SANDU, F. 2017. Network Automation and Abstraction using Python Programming Methods. *MACRo 2015*. https://doi.org/10.1515/macro-2017-0011

ORTEGA, J. M., SARKER, F., & WASHINGTON, S. 2019. *Learning Python Networking Second Edition A complete guide to build and deploy strong networking capabilities using Python 3.7 and Ansible* (2nd ed.). Packt Publishing.

RAHMAWATI. 2015. Konfigurasi Keamanan Jaringan Komputer Pada *Router* Dengan Metode ACL' S. *Teknik Komputer AMIK BSI*, *I*(2), 152–158. https://doi.org/10.31294/JTK.V1I2.246

RATAN, A. 2018. *Practical Network Automation - Second Edition* (2nd ed.). Packt Publishing. https://www.packtpub.com/networking-and-servers/practical-network-automation-second-edition

SWASTIKA, I. M. B., & ATITAMA, I. G. O. G. 2017. Otomatisasi Konfigurasi Mikrotik *Router* Menggunakan Software Ansible. In A. A. I. N. E. Karyawati, A. Z. Arifin, I. K. G. Suhartana, & A. Ashari (Eds.), *SNATIA (Seminar Nasional Teknologi Informasi & Aplikasinya)* (pp. 495–502).

WIJAYA, J. 2019. *Otomasi Jaringan dengan library Python dan Ansible untuk Pengelolaan Router Cisco* [Institut Teknologi Bandung]. https://digilib.itb.ac.id/index.php/gdl/view/39930/

Halaman ini sengaja dikosongkan