Jurnal Ilmu Komputer & Agri-Informatika
Volume 8 Nomor 2 halaman 126 – 134
e ISSN: 2654-9735, p ISSN: 2089-6026

Tersedia secara *online* di:
http://journal.ipb.ac.id/index.

Perbandingan Kinerja dan Kemudahan Implementasi Antara *Configuration Management* Ansible, Chef, dan Puppet

Comparison of Performance and Ease of Implementation Between Ansible, Chef, and Puppet Configuration Management

AURIZA RAHMAD AKBAR [1]*, JODHI LESMANA PUTRA [1]

Abstrak

Instalasi perangkat lunak lab komputer secara manual akan menyulitkan pemeliharaan. *Configuration management* (CM) dapat memudahkan instalasi komputer dalam jumlah besar dan kompleks secara konsisten serta menjaga sistem tetap dalam keadaan mutakhir. Penelitian ini bertujuan untuk membandingkan kinerja, sintaks, dan kemudahan implementasi beberapa CM yang populer: Ansible, Chef, dan Puppet. Pengujian dilakukan pada sebuah komputer di jaringan lokal menggunakan Vagrant dengan Virtualbox sebagai *provider*. Berdasarkan hasil pengujian pada tiga mesin virtual (VM) klien untuk proses instalasi aplikasi lab, Chef memerlukan waktu paling cepat (30 menit), disusul tipis oleh Ansible (31 menit), dan terakhir adalah Puppet (35 menit). Sintaks Chef paling disukai, sedangkan Ansible lebih mudah diimplementasikan karena tidak memerlukan server. Dari hasil tersebut, dapat disimpulkan bahwa CM yang cocok digunakan untuk skala lab komputer adalah Ansible karena kemudahan implementasi dengan kinerja dan dukungan yang baik.

Kata kunci**:** Ansible, Chef, *configuration management*, kinerja, Puppet

Abstract

Installing computer lab software manually makes maintenance costly. Configuration management (CM) could simplify big and complex computer installation consistently and keep the system updated. This research compare the performance, syntax, and ease of implementation of some popular CM: Ansible, Chef, and Puppet. The test was done on a computer in local network using Vagrant with Virtualbox as provider. The test results on three VM client for lab software installation process showed that Chef needs the least time (30 minutes), followed closely by Ansible (31 minutes), and then Puppet (35 minutes). Chef syntax is the most likeable, whereas Ansible is easier to implement because it doesn't need a server. From those results, we conclude that the right CM to be used in a computer lab scale is Ansible, because it is easier to implement with a good performance and support.

*Keywords**: Ansible, Chef, configuration management, performance, Puppet***

PENDAHULUAN

Instalasi perangkat lunak secara manual menambah kompleksitas pada sistem sehingga berdampak pada pemeliharaan yang sulit dan mahal. *Configuration management* (CM) bertujuan untuk mengatasi masalah tersebut dengan cara yang konsisten sehingga memudahkan instalasi komputer yang besar dan kompleks, serta menjaga sistem tetap dalam keadaan mutakhir (Torberntsson dan Rydin 2014). Penggunaan CM memungkinkan pendefinisian dan pengecekan perangkat lunak dilakukan secara serentak sehingga mengurangi kemungkinan kesalahan manusia (Fay *et al*. 2017). Selain itu, CM juga bersifat idempoten dengan menyimpan *state* modul, jika suatu modul sudah pernah dieksekusi sebelumnya maka tidak akan dieksekusi lagi pada iterasi selanjutnya (Hanappi *et al.* 2016).

Tersedia banyak pilihan CM yang membuat pengguna sulit memilih mana yang sesuai dengan kebutuhan mereka. Dua model utama dalam CM adalah *master-agent* dan *standalone*.

[1]Departemen Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Pertanian Bogor, Bogor 16680
*Penulis korespondensi. Surel: auriza@apps.ipb.ac.id

Pada model *master-agent*, terdapat server utama yang tersambung dengan klien dan memiliki basis data yang menyimpan data klien dan perangkat lunak yang terpasang. Pada model ini, terdapat skema *push* dan *pull*. Pada skema *pull*, agen yang terpasang pada klien akan tersambung dengan *master* secara periodik untuk mengecek konfigurasi terbaru. Sedangkan pada skema *push*, server yang akan memulai koneksi kepada klien untuk memberikan konfigurasi terbaru. Contoh CM yang menggunakan model ini adalah Chef dan Puppet. Sedangkan pada model *standalone*, klien tersambung langsung dengan *workstation* sehingga tidak memerlukan komputer server. Contoh CM yang menggunakan model ini adalah Ansible (Torberntsson dan Rydin 2014).

Tujuan dari penelitian ini adalah untuk membandingkan kinerja dan kemudahan implementasi CM berbasis Ansible, Chef, dan Puppet dalam instalasi dan konfigurasi perangkat lunak pada lab komputer Ilmu Komputer IPB. Batasan dari penelitian ini, yaitu:

- membandingkan tiga jenis CM: Ansible, Chef, dan Puppet
- konfigurasi komputer menyesuaikan kebutuhan lab Ilmu Komputer IPB
- sistem operasi Xubuntu 18.04 GNU/Linux
- konfigurasi instalasi memakai model *standalone*
- menggunakan server repositori lokal untuk mengunduh paket aplikasi

METODE

Penelitian ini dilakukan melalui beberapa tahap, yaitu identifikasi masalah yang akan diselesaikan, identifikasi kebutuhan perangkat keras dan perangkat lunak, implementasi, pengujian *script*, serta analisis hasil dan kinerja. Tahap penelitian tersebut dinyatakan pada Gambar 1.

Gambar 1 Tahapan penelitan.

Identifikasi Masalah

Setiap CM memiliki karakteristik dan fitur masing-masing. Karakteristik inilah yang membedakan kinerja dan implementasinya. Kurangnya penelitian dalam perbandingan kinerja inilah yang menyulitkan pengguna dalam memilih CM yang sesuai. Pada tahap ini, ditentukan CM yang akan diuji berdasarkan popularitasnya dari banyaknya jumlah pertanyaan yang berhubungan dengan kata kunci terkait di StackOverflow (2021). Hasilnya dapat dilihat pada Tabel 1, dengan tiga CM paling populer yaitu Ansible, Chef, dan Puppet.

Tabel 1 Kata kunci terkait CM pada StackOverflow

CM	Jumlah pertanyaan	Tahun awal rilis
Ansible	20581	2012
Chef	8250	2009
Puppet	3956	2005
Salt-stack	1269	2011

Identifikasi Kebutuhan

Paket aplikasi yang akan dipakai untuk pengujian pada penelitian ini mengikuti kebutuhan lab komputer Ilmu Komputer IPB. Daftar aplikasi yang akan diinstal dan dikonfigurasi pada tiap klien dapat dilihat pada Tabel 2. Spesifikasi mesin virtual (VM) klien yang digunakan untuk penelitian ini adalah: 1 vCPU Core i7-4770, vRAM 8 GB, dan vHDD 60 GB. Perangkat lunak yang digunakan untuk penelitian ini adalah: Ansible 2.7.4, Chef Workstation 0.2.48, Puppet 5.5.8, Ubuntu 18.04, Vagrant 2.1.2, dan Virtualbox 5.2.22.

Tabel 2 Daftar paket aplikasi lab yang digunakan dalam proses pengujian

Jenis	Nama paket aplikasi
`apt basic`	`tesseract-ocr-ara dia gimp graphviz libjpeg62 libavcodec-extra filezilla xul-ext-ublock-origin curl w3m unzip expect htop tmux itools redshift python-pip`
`apt CS lab`	`build-essential geany git yasm gcc-multilib gdb ddd clisp chezscheme openjdk-8-jdk logisim iverilog gtkwave fritzing fritzing-parts r-recommended python-opencv python3-opencv opencv-doc postgresql pgadmin3 pgcli sqlite3 apache2 postgis libjs-openlayers texlive texlive-publishers texlive-science pandoc pandoc-citeproc nmap traceroute whois mpi-default-bin mpi-default-dev openmpi-doc python3-pycryptodome pass geany-plugin-pg clustalw clustalx clustalo velvet soapdenovo2 bowtie2 ugene samtools plink1.9`
`dpkg`	`pencil.deb rstudio.deb`
`zip`	`postman.tar.gz pycharm.tar.gz gravit.zip android-studio.tar.gz geoserver.zip`
`expect`	`megan.sh`

Implementasi

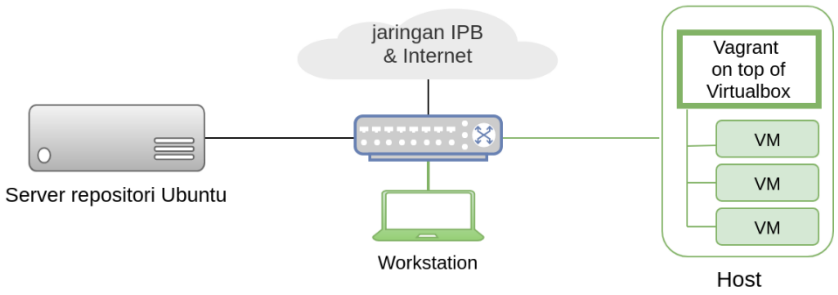
Implementasi Vagrant dimulai dengan instalasi Vagrant dengan menggunakan `apt`, kemudian instalasi Virtualbox sebagai *provider* utama pada Vagrant. Setelah itu, *box* Ubuntu untuk Vagrant diunduh dengan *provider* Virtualbox. Implementasi Ansible dimulai dengan menginstal paket Ansible dengan menggunakan `apt`. Kemudian mengunduh repositori Github yang berisi kode Ansible dan menjalankan *playbook* Ansible dengan menggunakan `ansible-playbook`. Implementasi Chef dimulai dengan mengunduh paket Chef-Workstation dari situs web Chef dan mengunduh repositori Github yang berisi kode Chef. Setelah itu paket diinstal dengan `dpkg`. *File* `solo.rb` dan *role* yang berisi konfigurasi Chef untuk berjalan secara lokal dibuat, kemudian *cookbook* dijalankan dengan menggunakan `chef-solo`. Implementasi Puppet dimulai dengan menginstal Puppet dengan `apt`. Repositori Github yang berisi kode Puppet diunduh, lalu menjalankan modul Puppet dengan `puppet apply`.

Pengujian

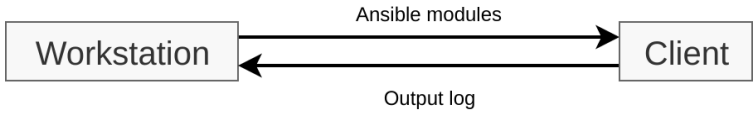
Pengujian dilakukan pada VM dengan sistem operasi Xubuntu 18.04. Komputer klien sudah terinstal SSH dengan alamat IP statis. Semua pengujian CM dilakukan dengan mode *standalone* tanpa server agar hasilnya seimbang. Pengujian Chef dilakukan menggunakan `chef-solo` dan dijalankan dua kali karena faktor *two pass model* untuk mencapai kondisi konvergen. Pengujian Puppet menggunakan `puppet-apply`. Pengujian Ansible tidak memerlukan komputer server, cukup menggunakan Python dan koneksi lokal.

Analisis Hasil dan Kinerja

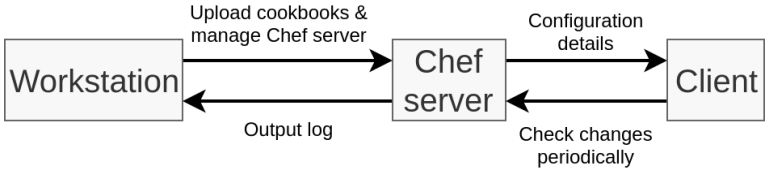
Analisis hasil dan kinerja dilakukan dari hasil yang didapatkan dari pengujian pada komputer *workstation*. Parameter uji yang digunakan berupa waktu instalasi dan konfigurasi pada VM klien. Jumlah klien yang diuji hingga tiga klien sekaligus, dengan jumlah ulangan masing-masing tiga kali. Skema pengujian kinerja dapat dilihat pada Gambar 2.



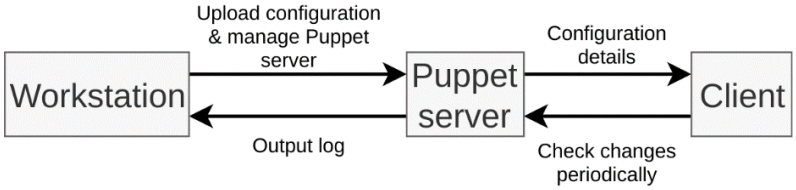
Gambar 2 Skema pengujian kinerja.



Gambar 3 Arsitektur komunikasi Ansible



Gambar 4 Arsitektur komunikasi Chef.



Gambar 5 Arsitektur komunikasi Puppet.

HASIL DAN PEMBAHASAN

Penelitian ini menggunakan Vagrant untuk manajemen VM. Konfigurasi Vagrant didefinisikan seperti kode di bawah ini untuk menentukan jenis sistem operasi, kapasitas penyimpanan, ukuran memori, dan jumlah CPU yang akan digunakan oleh VM. Setelah itu dilakukan instalasi Ansible, Chef, dan Puppet secara bergantian. Kode sumber dan konfigurasi lengkap dapat diakses pada repositori https://github.com/jodhi/skr.

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/bionic64"
  config.disksize.size = '60GB'
  config.vm.provider :virtualbox do |v|
    v.customize ["modifyvm", :id, "--memory", 8192]
    v.customize ["modifyvm", :id, "--cpus", 1]
  end
end
```

Perbandingan Arsitektur Komunikasi

Ansible terhubung dengan klien melalui koneksi SSH (Gambar 3). Komputer klien tidak perlu menginstal Ansible, cukup Python saja agar Ansible dapat menjalankan *playbook* (Hochstein 2015). Arsitektur *agentless* yang tidak memerlukan server terpusat ini juga membuat Ansible lebih unggul dari segi kemudahan dan keamanan (Ansible 2018).

Untuk Chef (Gambar 4), Chef-Server diperlukan pada server untuk mendukung fitur manajemen kebijakan, redundansi, dan basis data. Chef-Workstation diperlukan pada *workstation* untuk memasukkan konfigurasi ke server. Sedangkan pada klien, Chef-Client harus terinstal agar dapat terhubung ke server. Klien secara periodik akan menghubungi server untuk mengecek adanya perubahan pada *cookbooks* (Wettinger *et al.* 2013).

Arsitektur Puppet (Gambar 5) hampir sama seperti Chef. Puppet-Server diperlukan pada server untuk mendukung fitur manajemen kebijakan, redundansi, dan basis data. Puppet-Workstation juga diperlukan pada *workstation* untuk memasukkan konfigurasi ke server. Sedangkan pada klien, Puppet-Client harus terinstal agar dapat terhubung ke server. Klien secara periodik akan menghubungi server untuk mengecek adanya perubahan (Rhett 2016).

Perbandingan Sintaks

Konfigurasi repositori aplikasi.

Potongan kode pada Gambar 6, 7, dan 8 digunakan untuk mengubah konfigurasi repositori `apt` agar menggunakan alamat repositori yang diinginkan. Pada penelitian ini, repositori yang digunakan adalah repositori Ubuntu yang berada pada jaringan lokal lab CSN IPB. Pengubahan ini menggunakan fitur templat pada masing-masing CM.

Instalasi paket aplikasi dari repositori.

Untuk melakukan instalasi beberapa aplikasi sekaligus dari repositori dengan `apt`, semua CM menggunakan proses pengulangan pada suatu *list* (Gambar 9, 10, dan 11).

```
- name: configure local repo
  template:
    src: source.list
    dest: /etc/apt/sources.list
```

Gambar 6 Konfigurasi repositori pada Ansible.

```
template '/etc/apt/sources.list' do
  source 'source.list'
end
```

Gambar 7 Konfigurasi repositori pada Chef.

```
file { '/etc/apt/sources.list':
  ensure  => file,
  content => template('/home/vagrant/skr/puppet/templates/sources.list),
}
```

Gambar 8 Konfigurasi repositori pada Puppet.

```
- name: Install list of packages
  apt: name={{item}} state=present update_cache=true force_apt_get=yes
  with_items:
   - libreoffice
   - gimp
   - inkscape
```

Gambar 9 Instalasi aplikasi dari repositori pada Ansible.

```
package %w(libreoffice gimp inkscape) do
  action :install
end
```

Gambar 10 Instalasi aplikasi dari repositori pada Chef.

```
package {
  [libreoffice, gimp, inkscape]:
  ensure => 'installed'
}
```

Gambar 11 Instalasi aplikasi dari repositori pada Puppet.

Instalasi paket aplikasi dari luar repositori.

Ansible dan Chef dapat menggunakan fitur bawaan karena keduanya mendukung fitur *globbing* untuk *file*, berbeda dengan Puppet yang tidak mendukung *globbing*, sehingga dalam penelitian ini, digunakan *shell script* untuk Puppet. Potongan kode pada Gambar 12, 13, dan 14 digunakan untuk instalasi paket dari luar repositori.

```
- name: rstudio – list
  find:
    paths: ./
    patterns: "*rstudio*"
  register: rstudio_matched
- name: rstudio - install
  apt:
    deb: "{{ item.path }}"
    install_recommends: yes
  with_items:
    - "{{ rstudio_matched.files}}"
  ignore_errors: yes
```

Gambar 12 Instalasi paket dari luar repositori pada Ansible.

```
Dir['*rstudio*'].each do |path|
  dpkg_package 'install rstudio' do
    source path
    action :install
  end
end
```

Gambar 13 Instalasi paket dari luar repositori pada Chef.

```
exec { 'dpkg -i rstudio* ; apt-get install -f -y':
  cwd     => '/root',
  provider => 'shell'
}
```

Gambar 14 Instalasi paket dari luar repositori pada Puppet.

```
- name: Install postman – list
  find:
    paths: ./
    patterns: "Postman*"
  register: postman_matched
- name: Install postman - unzip
  command: tar -xf {{ item.path }} -C /opt
  become: true
  with_items:
    - "{{ postman_matched.files}}"
```

Gambar 15 Instalasi aplikasi dengan ekstraksi arsip pada Ansible.

```
Dir['Postman*'].each do |path|
  tar_extract "/"+path do
    action :extract_local
    target_dir '/opt/Postman'
    creates "/tmp/null"
  end
end
```

Gambar 16. Instalasi aplikasi dengan ekstraksi arsip pada Chef.

```
exec { 'tar -xf Postman*.gz -C /opt':
  cwd      => '/root',
  provider => 'shell'
}
```

Gambar 17 Instalasi aplikasi dengan ekstraksi arsip pada Puppet.

Instalasi aplikasi dengan ekstraksi arsip.

Ansible dan Puppet menggunakan *shell script* untuk melakukan ekstraksi arsip karena modul *default* yang tersedia tidak mendukung format .tar pada sistem operasi yang digunakan. Sedangkan Chef menggunakan modul `tar_extract` dari Chef Supermarket. Berikut contoh potongan kode untuk instalasi aplikasi Postman pada Gambar 15, 16, dan 17.

Instalasi aplikasi menggunakan Expect.

Ansible memiliki modul *default* yang mendukung jenis instalasi dan konfigurasi dengan Expect. Chef memiliki modul `expect` pada Chef Supermarket, namun modul tersebut tidak dapat bekerja dengan konsisten sehingga memakai *shell script* dengan templat seperti Puppet. Instalasi aplikasi menggunakan Expect dapat dilihat pada Gambar 18, 19, 20, dan 21.

```
- name: install megan with expect
  expect:
    command: ./MEGAN_*.sh
    responses:
      This will install: "\r"
      I accept the agreement: "1\r"
      Where should: "\r"
      Which components: "\r"
      Create symlinks: "\r"
      Select the folder: "\r"
      Check for updates: "\r"
      Set maximum allowed memory: "\r"
      Run MEGAN: "n\r"
```

Gambar 18 Instalasi aplikasi dengan Expect pada Ansible.

```
set timeout -1
spawn /root/MEGAN_*.sh
expect "This will install"          {send "\r"}
expect "I accept the agreement"     {send "1\r"}
expect "Where should"               {send "\r"}
expect "Which components"           {send "\r"}
expect "Create symlinks"            {send "\r"}
expect "Select the folder"          {send "\r"}
expect "Check for updates"          {send "\r"}
expect "Set maximum allowed memory" {send "\r"}
expect "Run MEGAN"                  {send "n\r"}
expect eof
```

Gambar 19 Templat instalasi aplikasi dengan Expect berupa *shell script* (`expect_megan.sh`).

```
template '/expect_megan.sh' do
  source 'expect_megan.sh'
  mode '0755'
end

execute 'Install megan (expect)' do
  command "expect -f /expect_megan.sh"
  creates "/opt/megan"
end
```

Gambar 20 Instalasi aplikasi dengan Expect pada Chef.

```
class install_megan {
  file { '/root/expect_megan.sh':
    ensure  => file,
    content => template('/home/vagrant/skr/puppet/templates/expect_megan.sh'),
  }

  exec { 'expect -f /root/expect_megan.sh':
    cwd      => '/root',
    provider => 'shell'
  }
}
```

Gambar 21 Instalasi aplikasi dengan Expect pada Puppet.

Tabel 3 Survei kemudahan sintaks CM

Jenis sintaks	Jumlah pilihan		
	Ansible	Chef	Puppet
Konfigurasi repositori	6	4	0
Instalasi paket aplikasi repositori	1	3	6
Instalasi paket aplikasi di luar repositori	1	7	2
Instalasi aplikasi dengan ekstraksi arsip	3	4	3
Instalasi aplikasi dengan Expect	6	4	0
Total	17	22	13

Tabel 4 Waktu proses CM dalam proses instalasi aplikasi lab (menit:detik)

Jumlah klien	Ansible	Chef	Puppet
1	20:57	20:22	24:26
2	27:21	25:59	29:35
3	31:22	29:50	34:54

Hasil survei dari 10 responden mahasiswa Ilmu Komputer IPB tentang sintaks CM dapat dilihat pada Tabel 3. Sintaks Chef yang berbasis dari bahasa pemrograman Ruby paling disukai, karena pengguna merasa lebih ringkas dan mudah dibaca. Sedangkan sintaks Ansible yang memakai format YAML, meskipun mudah dibaca namun memerlukan lebih banyak baris kode. YAML memerlukan baris baru sebagai pembatas antar-*item*. Adapun untuk Puppet yang memiliki bahasa tersendiri, mungkin sintaksnya kurang familiar bagi kebanyakan pengguna.

Perbandingan Kinerja

Waktu pemrosesan tiap CM pada Tabel 4 didapatkan dari rataan waktu pengujian sebanyak tiga kali ulangan untuk masing-masing jumlah VM klien. Skema pengujian dapat dilihat pada Gambar 2.

Pada pengujian Chef terbagi menjadi dua bagian utama yaitu *compile* dan *converge*. Hal ini dikarenakan model *two-pass* pada Chef. *Compile* dilakukan saat inisiasi awal berupa melakukan konfigurasi selain *keyword* `ruby_block`, `lazy`, dan `only_if` dan kemudian *converge* dimaksudkan untuk dilakukan secara berkali-kali yang sesuai dengan desain awal Chef yang melakukan pengecekan *state* setiap interval tertentu.

Pada Puppet, instalasi yang memerlukan *globbing* perlu menggunakan *shell script* karena Puppet tidak mendukung evaluasi variabel dinamis seperti Ansible dan Chef. Ansible mendukung fitur tersebut dengan cara mengevaluasi variabel yang sesuai dengan pola yang diinginkan, lalu menyimpan nama dan *path* berkas yang sesuai pada sebuah variabel. Sedangkan pada Chef, hal ini dilakukan dengan cara mengevaluasi pola pada saat *compile*.

Dari sisi implementasi, Ansible merupakan yang termudah karena tidak membutuhkan komputer tambahan sebagai server, tidak seperti Chef dan Puppet yang memerlukan server untuk manajemen konfigurasi.

SIMPULAN

Hasil pengujian pada klien VM menunjukkan bahwa waktu proses Chef adalah yang tercepat, disusul tipis oleh Ansible. Hasil survei pengguna juga menunjukkan sintaks CM yang paling disukai adalah Chef. Sedangkan dari sisi kemudahan implementasi, Ansible unggul karena tidak membutuhkan server, serta paling populer sehingga mudah menemukan bantuan. Dari hasil tersebut, dapat disimpulkan bahwa CM yang tepat digunakan dalam skala lab komputer adalah Ansible karena mudah dalam implementasi dengan kinerja yang baik. Penelitian lebih lanjut diperlukan untuk mengukur kinerja CM secara nyata pada komputer fisik dengan jumlah yang banyak.

DAFTAR PUSTAKA

Ansible. 2018. The benefits of agentless architecture. [diakses 2021 Nov 27]. https://ansible.com/hubfs/pdfs/Benefits-of-Agentless-WhitePaper.pdf.

Fay R, Bland J, Jones S. 2017. Next generation monitoring: tier 2 experience. Di dalam: *J Phys: Conf Ser*. 898(9):092035. IOP. doi: 10.1088/1742-6596/898/9/092035.

Hanappi O, Hummer W, Dustdar S. 2016. Asserting reliable convergence for configuration management scripts. Di dalam: *OOPSLA*. Amsterdam: ACM. hlm 328–343. doi: 10.1145/2983990.2984000.

Hochstein L. 2015. *Ansible: Up & Running*: *Automating Configuration Management and Deployment The Easy Way*. Sebastopol (CA): O'Reilly.

Rhett J. 2016. *Learning Puppet 4: A Guide to Configuration Management and Automation*. Sebastopol (CA): O'Reilly.

StackOverflow. 2021. Tags. [diakses 2021 Nov 27]. https://stackoverflow.com/tags.

Torberntsson K, Rydin Y. 2014. A study of configuration management systems: Solutions for deployment and configuration of software in a cloud environment. Uppsala: Uppsala Universitet.

Wettinger J, Behrendt M, Binz T, Breitenbücher U, Breiter G, Leymann F, Moser S, Schwertle I, Spatzier T. 2013. Integrating configuration management with model-driven cloud management based on TOSCA. Di dalam: *CLOSER*. Stuttgart: SciTePress. hlm 437–446.