PENGUJIAN ANSIBLE PLAYBOOK MENGGUNAKAN *MOLECULE* DAN *GITHUB ACTIONS*

Desy Intan Permatasari[1], Andika Ahmad Ramadhan[2]
[1,2] Politeknik Elektronika Negeri Surabaya,
Departemen Teknik Informatika dan Komputer, Program Studi Teknik Informatika,
email: [1]desy@pens.ac.id, [2]andikahmadr@gmail.com

ABSTRAK

Pengujian Ansible Playbook masih jarang dilakukan oleh kebanyakan pengguna Ansible, sehingga dapat menyebabkan kegagalan yang tidak terduga saat menjalankan playbook tersebut. Hal ini menyebabkan kerugian yang cukup signifikan terhadap sebuah organisasi atau instansi yang menggunakan Ansible sebagai tonggak utama untuk menjalankan proses otomasi tugas. Pengujian dengan sebuah ansible role menggunakan molecule, dapat membantu dalam proses development maupun proses deployment sebuah role. *Compatibility testing* pada ansible role yang telah dibuat, dapat mengetahui kompatibilitasnya pada sistem yang berbeda-beda. Hasil dari penelitian ini dapat digunakan sebagai acuan untuk melakukan testing pada sebuah Ansible Playbook, sehingga terjadinya kesalahan saat menjalankan playbook menjadi terminimalisir.

Kata Kunci: *Ansible Playbook, Molecule, Compatibility Testing*.

ABSTRACT

Testing of the Ansible Playbook is still rarely performed by most Ansible users, so it can cause unexpected failures when running the playbook. This causes a significant loss to an organization or agency that uses Ansible as the main milestone to run the task automation process. Testing with an ansible role using a molecule can help in the development process as well as the deployment process for a role. Compatibility testing on ansible roles that have been created, can find out their compatibility on different systems. The results of this study can be used as a reference for testing an Ansible Playbook, so that errors when running the playbook are minimized.

Keywords: *Ansible Playbook, Molecule, Compatibility Testing*.

PENDAHULUAN

Dalam urutan proses pembangunan perangkat lunak, pengujian perangkat lunak adalah tahap yang dilakukan setelah implementasi rancangan perangkat lunak kedalam bentuk kode. Pengujian perangkat lunak atau software testing adalah proses menguji program dengan maksud mencari kesalahan awal sebelum program diberikan ke pengguna. Pada saat ini setiap setelah melakukan pembangunan atau pengembangan perangkat lunak akan selalu dilakukan pengujian perangkat lunak dengan teknik yang berbeda-beda.

Pengujian *Ansible Playbook* masih jarang dilakukan oleh kebanyakan pengguna *Ansible*, sehingga dapat menyebabkan kegagalan yang tidak terduga saat menjalankan *playbook* tersebut. Hal ini menyebabkan kerugian yang cukup signifikan terhadap sebuah organisasi atau instansi yang menggunakan *Ansible* sebagai tonggak utama untuk menjalankan proses otomasi tugas. Penelitian ini bertujuan untuk mengimplementasikan proses testing pada sebuah *Ansible Playbook* dengan menggunakan *Molecule*. Hasil dari penelitian ini dapat digunakan sebagai panduan untuk melakukan testing pada sebuah *Ansible Playbook*, sehingga terjadinya kesalahan saat menjalankan *playbook* menjadi terminimalisir.

 ISSN: 2598-0076

METODE

Definisi *Software Testing* / Pengujian Perangkat Lunak menurut standar ANSI / IEEE 1059 adalah sebuah proses menganalisis item perangkat lunak untuk mendeteksi perbedaan antara kondisi yang ada dan yang diperlukan (yaitu, cacat) dan untuk mengevaluasi fitur item perangkat lunak.

Tidak semua error pada perangkat lunak disebabkan oleh kesalahan dalam penulisan kode program. Salah satu sumber umum dari error adalah perbedaan persyaratan, yaitu, persyaratan yang tidak dikenal yang mengakibatkan kesalahan kelalaian oleh perancang program. Perbedaan persyaratan seringkali dapat berupa persyaratan non-fungsional seperti kemampuan pengujian, skalabilitas, pemeliharaan, kinerja, dan keamanan.

Kesalahan perangkat lunak terjadi melalui proses berikut. Seorang *programmer* membuat kesalahan (kesalahan), yang mengakibatkan cacat (kesalahan, bug) pada kode sumber perangkat lunak. Jika cacat ini dijalankan, dalam situasi tertentu sistem akan menghasilkan hasil yang salah sehingga menyebabkan kegagalan. [6] Tidak semua cacat akan menyebabkan kegagalan. Misalnya, cacat pada kode mati tidak akan pernah mengakibatkan kegagalan. Cacat dapat berubah menjadi kegagalan jika lingkungan diubah. Contoh perubahan lingkungan ini termasuk perangkat lunak yang dijalankan pada platform perangkat keras komputer baru, perubahan dalam sumber data, atau berinteraksi dengan perangkat lunak yang berbeda. Cacat tunggal dapat menyebabkan berbagai gejala kegagalan.

Strategi *software testing* dapat dilakukan secara manual atau otomatis. Menurut strategi pengujian manual, pendekatan yang lebih tradisional, penguji menyiapkan *suite* tes yang menurut mereka akan lebih baik menjalankan program. Alat pengujian perangkat lunak otomasi membantu dalam menghasilkan kasus uji dari spesifikasi program atau dari data teks aslinya. Pengujian manual adalah teknik pengujian di mana insinyur penguji menyiapkan test case secara manual dan mengeksekusinya untuk mengidentifikasi cacat pada perangkat lunak. Mengotomatiskan pengujian perangkat lunak menggunakan bahasa scripting seperti *Python, JavaScript* atau *Tool Command Language* karena kasus uji dapat dengan mudah dieksekusi oleh mesin dengan sedikit intervensi manusia dan perhatian.

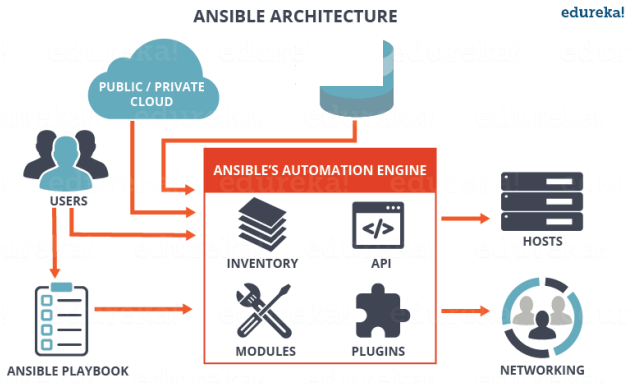
Ansible adalah sebuah perangkat lunak yang digunakan untuk melakukan *provisioning* yang dikembangkan oleh RedHat. *Ansible* dapat digunakan untuk membantu seorang *system administrator* dalam pekerjaan untuk melakukan *setup* maupun konfigurasi kepada sebuah *server*. *Ansible* dikembangkan menggunakan bahasa pemrograman *Python*.

Hingga saat ini *Ansible* sudah mencapai versi 2.9. Karena dibuat menggunakan Bahasa pemrograman *python*, *Ansible* juga melakukan proses *provisioning* dengan menggunakan bantuan python. *Ansible* berjalan secara *agentless* atau tanpa menggunakan *agent* (layanan yang dipasang pada system target).

 ISSN: 2598-0076

Syarat dari sebuah system atau *server* dapat dikonfigurasi menggunakan *Ansible* adalah sudah terpasang python disana. *Ansible* menggunakan layanan SSH *(Secure Shell)* untuk dapat berkomunikasi dengan sistem tujuan. Jadi untuk menggunakan layanan *Ansible* tidak membutuhkan banyak *dependency* yang terpasang.

Diagram dibawah ini menjelaskan bagaimana *Ansible* dapat melakukan konfigurasi pada *server* target.



Gambar 1. Arsitektur Ansible

Ansible Automation Engine adalah semua komponen utama dari *Ansible* berada. Beberapa komponen utama dari *Ansible* adalah *inventory, modules*, API, dan *Plugin*. *User* adalah seorang pengguna atau system administrator yang akan menjalankan *Ansible*. Pengguna dapat langsung menggunakan *Ansible* di perangkat miliknya sendiri. Ansible juga dapat dipasang sebagai layanan yang terpusat, sehingga manajemen penggunaan *Ansible* menjadi lebih mudah dengan menggunakan *Ansilbe Tower*, *Ansible AWX*, atau *Ansible CMDB.*

Ansible CMDB adalah sebuah layanan yang menggunakan tampilan web sebagai tempat untuk melakukan kontrol terhadap penggunaan Ansible. Didalam CMDB dapat menyimpan informasi dan melihat status dari hosts yang dimiliki. *Ansible Playbook* memungkinkan untuk membuat sebuah list dari perintah-perintah yang akan dijalankan oleh Ansible. *Playbook* pada umumnya memiliki format file yaml. Pengguna bisa menuliskan langkah-langkah yang dibutuhkan untuk dapat menyelesaikan sebuah tugas. Untuk menjalankan sebuah perintah Ansible bisa menggunakan modules tertentu melalui terminal atau melalui playbook. Hosts adalah kumpulan dari server tujuan yang sudah didefinisikan dan dikelompokkan sesuai dengan kebutuhan pengguna. Networking adalah sebuah modules yang tersedia pada Ansible untuk melakukan konfigurasi pada perangkat jaringan.

HASIL DAN PEMBAHASAN

Pengujian dapat dilakukan dengan melakukan beberapa tahap berikut:

a.) Menyiapkan *Ansible Role* dengan *Molecule* dengan cara menginstall *Molecule*

 ISSN: 2598-0076

```
pip install molecule
```

b) Mengecek apakah *molecule* sudah terpasang

```
$ molecule --version
molecule, version 2.19.0
```

c) Membuat *ansible role* baru dengan *molecule*

```
molecule init role geerlingguy.example -d docker
```

Perintah tersebut menggunakan ansible-galaxy di belakang layar untuk menghasilkan Ansible Role baru, Di dalam direktori Molecule adalah direktori default, yang menunjukkan skenario pengujian default seperti berikut:

```
$ cd geerlingguy.example/molecule/default/ && ls
Dockerfile.j2
INSTALL.rst
molecule.yml
playbook.yml
tests
```

d) Membuat sebuah role yang dapat melakukan instalasi *apache2* kedalam beberapa jenis Sistem Operasi atau Distro Linux

```
# kmdr7 @ ssl-u12-l41f in /work/code/ansible/ansible-role-apache
$ tree
.
├── defaults
│   └── main.yml
├── handlers
│   └── main.yml
├── LICENSE
├── meta
│   └── main.yml
├── molecule
│   └── default
│       ├── converge.yml
│       └── molecule.yml
├── README.md
├── tasks
│   ├── configure-Debian.yml
│   ├── configure-RedHat.yml
│   ├── configure-Solaris.yml
│   ├── configure-Suse.yml
│   ├── main.yml
│   ├── setup-Debian.yml
│   ├── setup-RedHat.yml
│   ├── setup-Solaris.yml
│   └── setup-Suse.yml
├── templates
│   └── vhosts.conf.j2
└── vars
    ├── AmazonLinux.yml
    ├── apache-22.yml
    ├── apache-24.yml
    ├── Debian.yml
    ├── RedHat.yml
    ├── Solaris.yml
    └── Suse.yml
```

 ISSN: 2598-0076

e) Jalankan testing dengan *molecule*

```
$ cd ../../
$ molecule test
...
--> Validating schema /Users/jgeerling/Downloads/geerlingguy.example/molecule/default/molecule.yml.
Validation completed successfully.
--> Test matrix

└── default
    ├── lint
    ├── destroy
    ├── dependency
    ├── syntax
    ├── create
    ├── prepare
    ├── converge
    ├── idempotence
    ├── side_effect
    ├── verify
    └── destroy

--> Executing Yamllint on files found in /Users/jgeerling/Downloads/geerlingguy.example/...
Lint completed successfully.
--> Action: 'syntax'
    playbook: /Users/jgeerling/Downloads/geerlingguy.example/molecule/default/playbook.yml
--> Action: 'converge'
--> Action: 'idempotence'
Idempotence completed successfully.
...
```

Molekul menjalankan semua langkah pengujian: linting, memeriksa sintaks *playbook*, membangun lingkungan *Docker*, menjalankan playbook di lingkungan *Docker*, menjalankan *playbook* lagi untuk memverifikasi idempotensi, lalu membersihkannya sendiri

f) Gunakan *docker image* yang sudah dibuat khusus untuk melakukan testing dengan molecule

```
platforms:
  - name: instance
    image: "geerlingguy/docker-${MOLECULE_DISTRO:-centos7}-ansible:latest"
    command: ${MOLECULE_DOCKER_COMMAND:-""}
    volumes:
      - /sys/fs/cgroup:/sys/fs/cgroup:ro
    privileged: true
    pre_build_image: true
```

Konfigurasi dapat diubah di file molecule.yaml. Banyak opsi lainnya, seperti *volume, command, image*, dan *privileges* yang di parse kedalam sistem *docker compose*, atau dalam parameter modul Ansible *docker_container*. Opsi khusus *pre_build_image*, jika disetel ke *true*, berarti *image* yang Anda gunakan sudah memiliki *Ansible* di dalamnya, jadi *Molecule* tidak perlu membuang waktu untuk membuatnya jika gambar tersebut tidak ada secara lokal. Mungkin juga memperhatikan penggunaan variabel seperti $ {MOLECULE_DISTRO: -centos7}; *Molecule* dengan mudah mendukung variabel lingkungan di dalam file *molecule.yml*, dan Anda bahkan dapat memberikan default seperti yang saya miliki dengan sintaks: -centos7 (ini berarti bahwa jika MOLECULE_DISTRO adalah string kosong atau tidak disetting, maka akan default ke centos7).

g) Melakukan integrasi *Molecule* dengan Travis CI

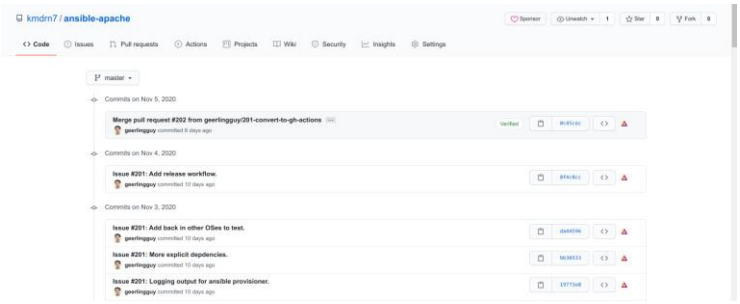
Dalam *Jenkins role* (dan banyak lainnya), penulis menentukan *MOLECULE_PLAYBOOK* yang berbeda untuk guide dari konvergensi. Beberapa orang mungkin lebih menyukai skenario molekul yang sama sekali berbeda (selain skenario default) untuk kasus pengujian yang berbeda, tetapi pada kasus ini, penulis menetapkan nama buku pedoman konvergen (*provisioner.playbooks.converge* di dalam molekul.yml) ke variabel lingkungan sehingga penulis bisa menggunakan pedoman yang berbeda untuk setiap skenario:

 ISSN: 2598-0076

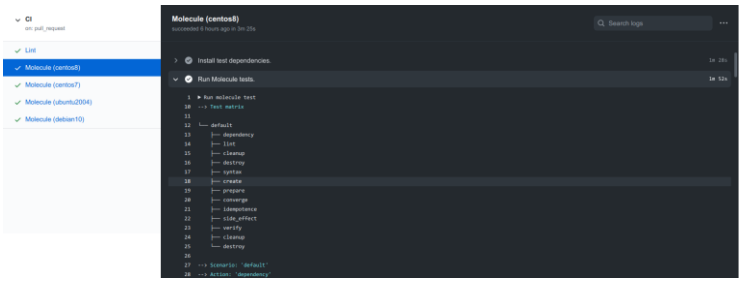
```
provisioner:
  name: ansible
  lint:
    name: ansible-lint
  playbooks:
    converge: ${MOLECULE_PLAYBOOK:-playbook.yml}
```

h) Menyiapkan *Github Action* dan deploy ke *Github*

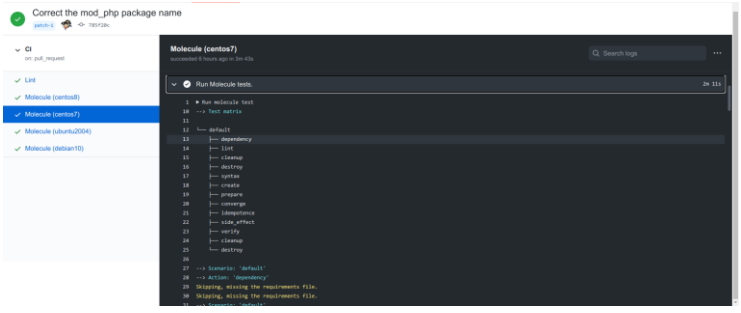
i) Melakukan push pada *repository* yang telah dikonfigurasi



j) *Github Actions* akan mendeteksi sebuah push baru dan akan menjalankan otomasi testing ansible role

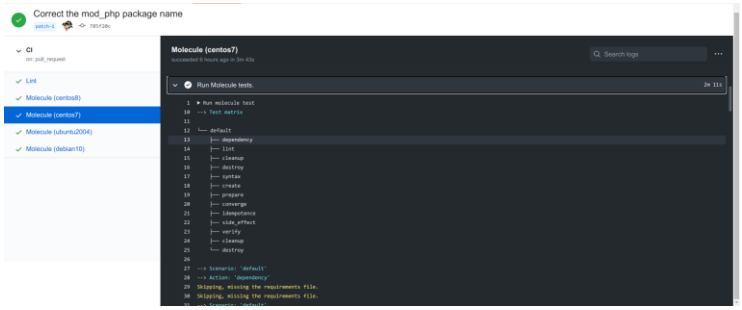


Gambar 2. Role telah berjalan di distro Centos8

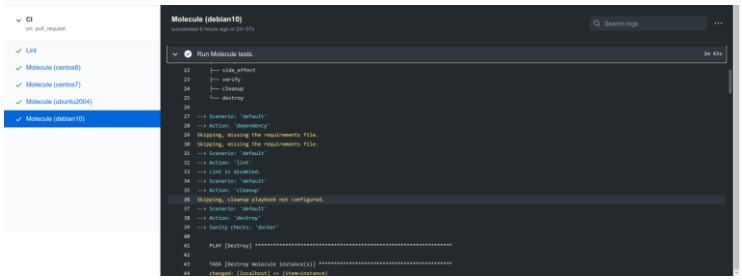


Gambar 3. Role telah berjalan di distro Centos7

ISSN: 2598-0076



Gambar 4. Role telah berjalan di Ubuntu 20.04



Gambar 5. Role telah berjalan di Debian10

Dengan melakukan pengujian sebuah *ansible role* menggunakan *molecule*, dapat membantu dalam proses development maupun proses *deployment* sebuah role. Dengan menerapkan *compatibility testing*, maka *ansible role* yang telah dibuat dapat diketahui kompatibilitasnya jika akan dipasang pada sistem yang berbeda-beda.

Molecule juga merupakan definisi terbaru dari *"Testing Ansible Role"* dikarenakan hingga saat molecule baru muncul, para developer role masih menggunakan cara testing yang manual. *Developer* masih membuat sebuah alat yang digunakan untuk melakukan otomasi *ansible role* ini. Yang menjadi kendala utama adalah alat tersebut masih belum ada standardnya, sehingga menyebabkan kurangnya kemudahan untuk *developer* yang baru saja belajar membuat sebuah *ansible role* menjadi dimudahkan.

SIMPULAN

Pengujian dengan sebuah ansible role menggunakan molecule, dapat membantu dalam proses development maupun proses deployment sebuah role. *Compatibility testing* pada ansible role yang telah dibuat, dapat mengetahui kompatibilitasnya pada sistem yang berbeda-beda. Molecule juga merupakan definisi terbaru dari *"Testing Ansible Role"* dikarenakan hingga saat molecule baru muncul, para *developer role* masih menggunakan cara testing yang manual. *Developer* masih membuat sebuah alat yang digunakan untuk melakukan otomasi ansible role ini. Kendala yang ditemui adalah alat tersebut masih belum ada standardnya, sehingga developer yang baru mempelajari dalam membuat sebuah *ansible role* akan menemui kesulitan. Hasil dari penelitian ini dapat digunakan sebagai acuan untuk melakukan testing pada sebuah *Ansible Playbook*, sehingga terjadinya kesalahan saat menjalankan *playbook* menjadi terminimalisir

 ISSN: 2598-0076

DAFTAR RUJUKAN

I Made Bayu Swastika , I Gede Oka Gartria Atitama, "*Otomatisasi Konfigurasi Mikrotik Router Menggunakan Software Ansible, Seminar Nasional Teknologi Informasi dan aplikasinya."*

Donny Rahardika , Niki Ratama, *"Implementasi Network Automation Untuk Konfigurasi Jaringan Baru Dengan Netmiko, Journal of Artificial Intelligence and Innovative Applications"*. Vol. 2, No. 3, Agustus 2021l

M. F. Islami, P. Musa, and M. Lamsani, *"Implementation of Network Automation using Ansible to Configure Routing Protocol in Cisco and Mikrotik Router with Raspberry PI,"* J. Ilm. Komputasi, vol. 19, no. 2, pp. 127–134, 2020, doi: 10.32409/jikstik.19.2.80.

S. Yadav, *"Research Paper on Network Automation,"* Int. J. Res. Appl. Sci. Eng. Technol., vol. 7, pp. 1446–1450, Apr. 2019, doi: 10.22214/ijraset.2019.4261.s

 ISSN: 2598-0076