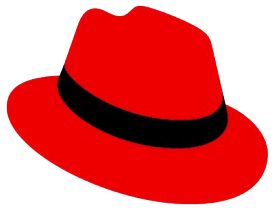
Red Hat Summit

Connect

Test Driven Ansible

Ein Experiment auf dem Weg zur robusteren Automation

Red Hat



Red Hat

Steffen Frömer

Technical Account Manager
Red Hat

Red Hat

Question time

We’ve asked 100 people ...

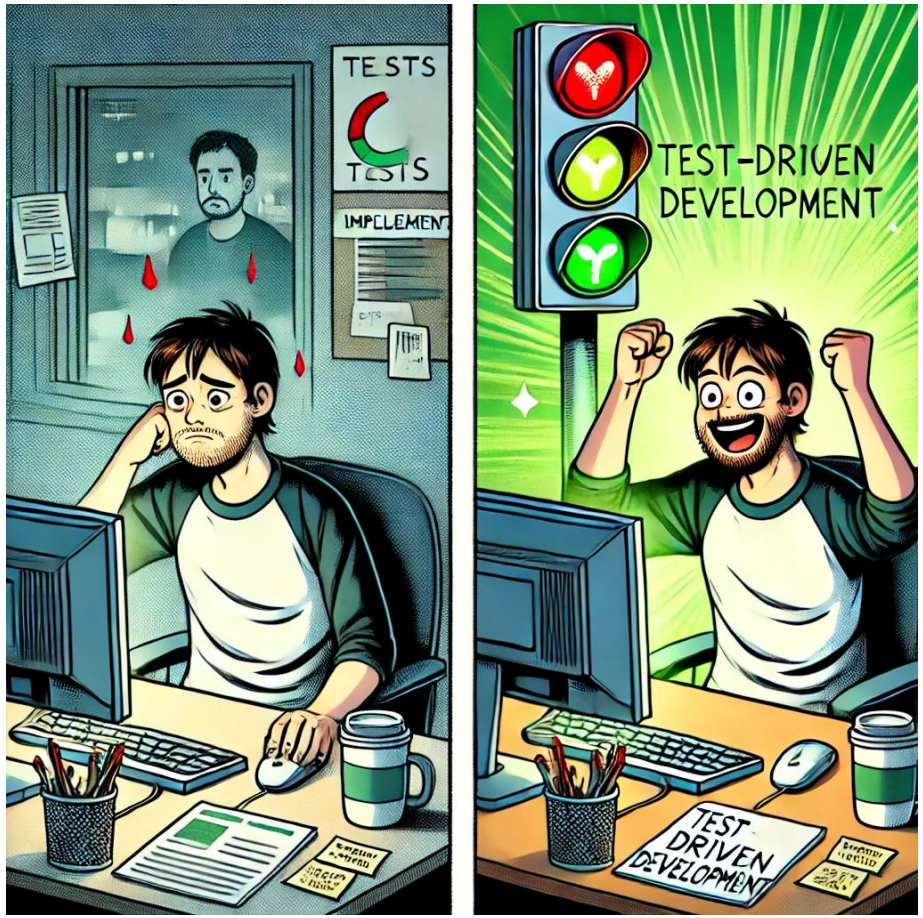
Motivation

Let me tell you a story ...

- I'm a Sysadmin
- I can code, but not develop
- Creating tests is boring and waste of time

Motivation

Let me tell you a story ...

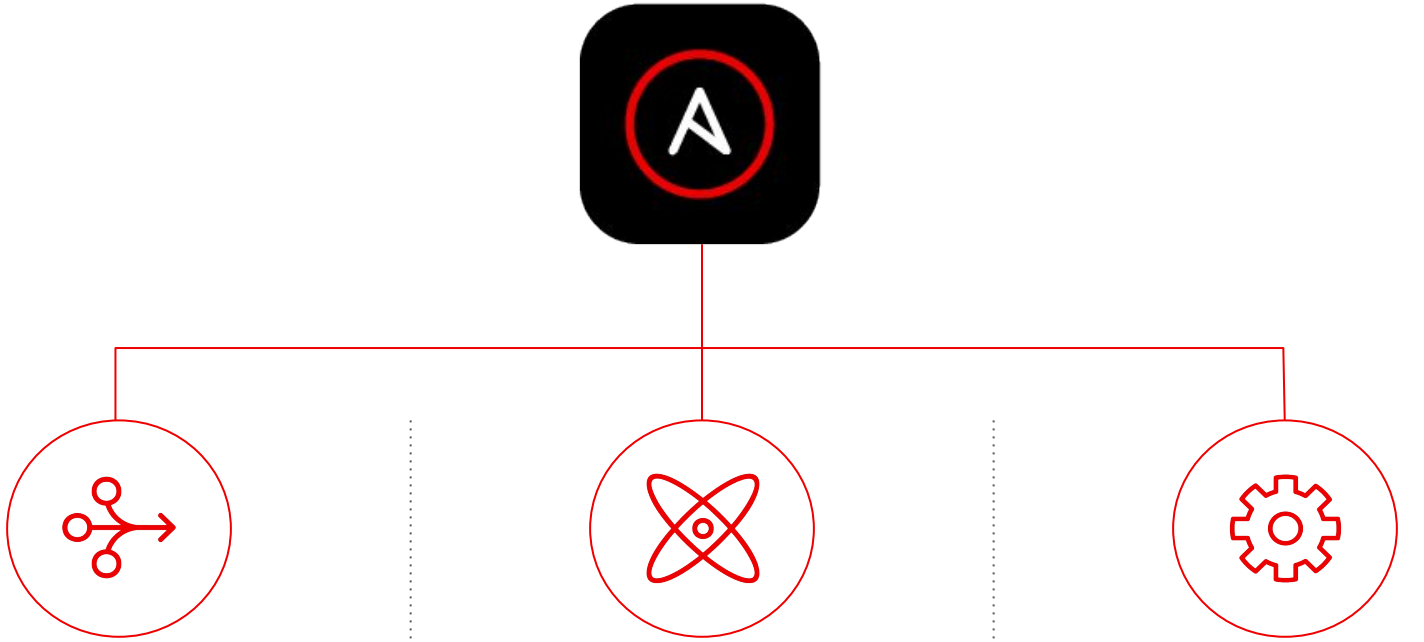


- I'm a Sysadmin
- I can code, but not develop
- Creating tests is boring and waste of time

- Curious about TDD
- Used approach to develop a parser
- Run into challenge where complete refactor was necessary
- It went well without breaking anything

Red Hat

Why using Ansible?



Simple

Human-readable language with quick adoption

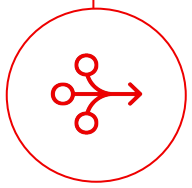
Powerful

Universal language spanning multiple IT domains

Agentless

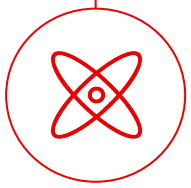
Easily integrate with hybrid environments

Why using Ansible?



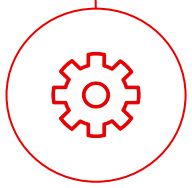
Simple

Human-readable language with quick adoption



Powerful

Universal language spanning multiple IT domains



Agentless

Easily integrate with hybrid environments

- Simplifying Infrastructure Management
- Streamlining configuration management
- improved Efficiency and Productivity
- Scalability and Flexibility
- Infrastructure as Code
- Security and Compliance

You need testing to fully realize these benefits

Ways of testing

- Ansible Lightspeed

webserver.yml U

webserver.yml

```yaml
---
- name: Install and configure webserver on RHEL box.
  hosts: webservers
  become: yes
  gather_facts: yes

  tasks:
    - name: Install and configure webserver on RHEL box.
      when: ansible_os_family == "RedHat"
      ansible.builtin.package:
        name: httpd
        state: present

    # Now open the port for the webserver.
    - name: Now open the port for the webserver
      ansible.posix.firewalld:
        port: 80/tcp
        permanent: true
        state: enabled
        immediate: true

    # Next, put a file in the webservers directory provided as index.html
    - name: Next, put a file in the webservers directory provided as index.html
      ansible.builtin.copy:
        src: files/index.html
        dest: /var/www/html/index.html
        owner: root
        group: root
        mode: '0644'

    # Finally, ensure selinux is not blocking the webserver
    - name: Finally, ensure selinux is not blocking the webserver
      ansible.posix.seboolean:
        name: httpd_can_network_connect
        state: true
        persistent: true

    # And make sure the services is running
    - name: And make sure the services is running
      ansible.builtin.service:
        name: httpd
        state: started
        enabled: true

```

Ways of testing

- Ansible Lightspeed
- ansible-playbook *--syntax-check*

```
$ ansible-playbook -i inventory --syntax-check
webserver.yml

playbook: webserver.yml
$
```

Ways of testing

- Ansible Lightspeed
- ansible-playbook *--syntax-check*
- ansible-lint

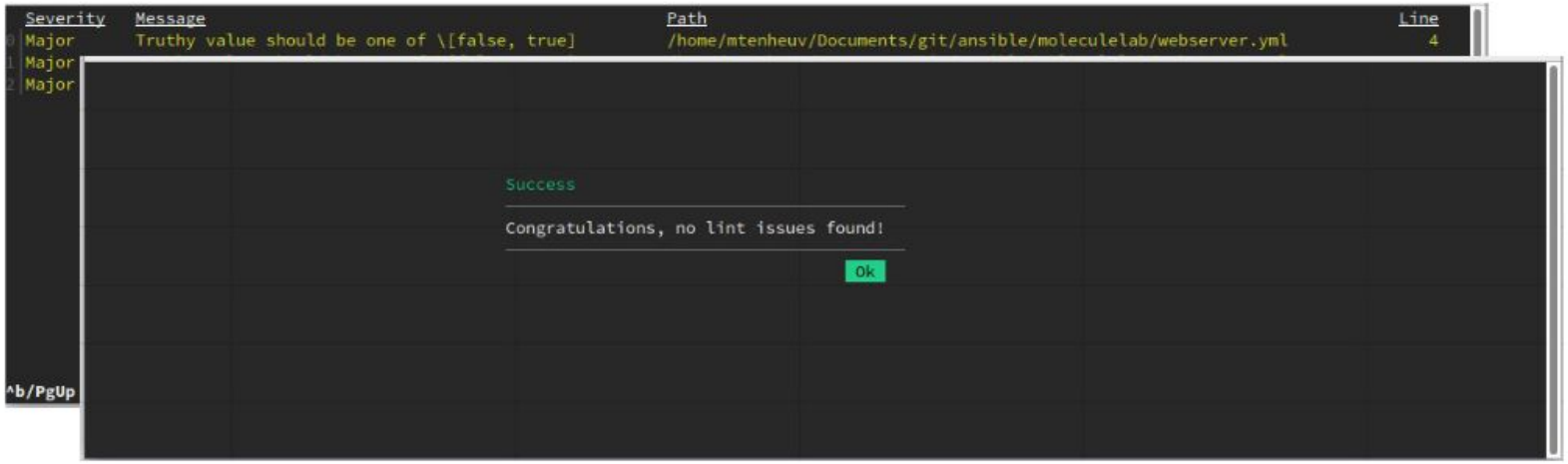
```
$ ansible-lint --offline -p webserver.yml

Passed: 0 failure(s), 0 warning(s) on 1 files. Last
profile that met the validation criteria was
'production'.

$
```

Ways of testing

- Ansible Lightspeed
- ansible-playbook *--syntax-check*
- ansible-lint
- ansible-navigator



Red Hat

Ways of testing

- Ansible Lightspeed
- ansible-playbook *--syntax-check*
- ansible-lint
- ansible-navigator

Is that everything? Feels odd, there have to be better ways to perform testing.

Ansible molecule

Consistent, repeatable testing for Ansible content

- Molecule provides support for testing with multiple instances, operating systems and distributions, virtualization providers, test frameworks and testing scenarios.
- Molecule encourages an approach that results in consistently developed roles that are well-written, easily understood and maintained.

Molecule key concepts

What are the main components and functionalities?

Building

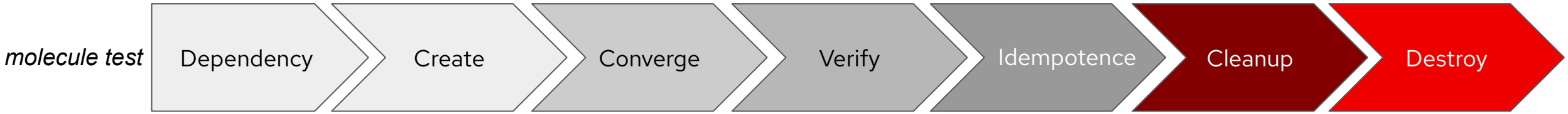
- Use a ***molecule "scenario"*** to define a customizable and modular sequence of test steps
- Use ***molecule init scenario*** to initialize the scenario while developing your collection
- Define an ***ephemeral test environment*** based on a selected driver (on cloud, podman, ocp ...) or custom driver

Testing

- Create/cleanup/destroy/re-create the ephemeral test environment
- ***Test roles or collection*** playbooks by developing your own "functional" tests
- Include playbook ***idempotence test*** and ***syntax check***

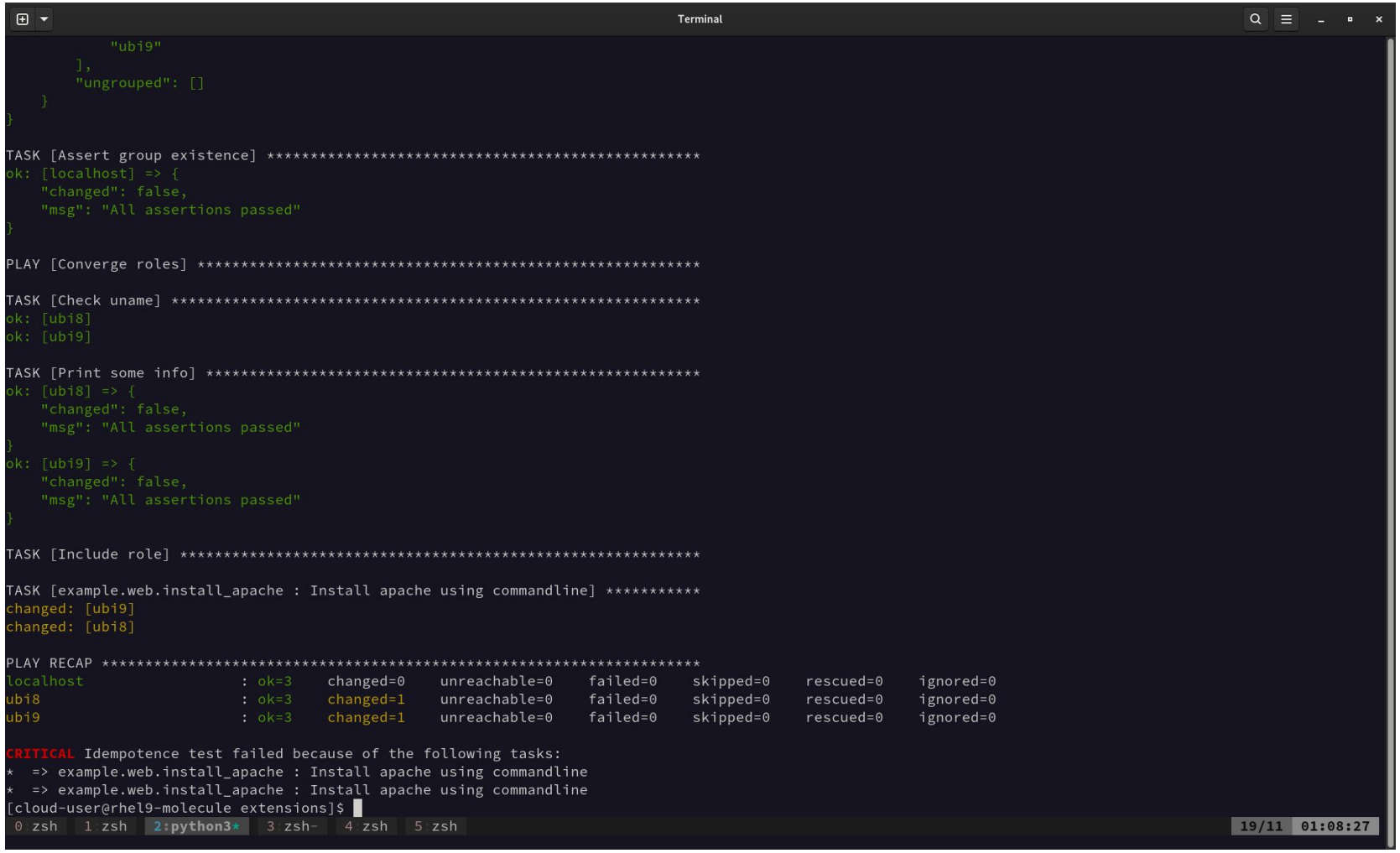
Molecule typical run

What happens when we run molecule?



- This is a typical sequence that can be expanded or modified based on the needs
- Each step can be executed individually (molecule <<step name>>)
- Steps are defined as individual files in yml format (“as a code”)
- A main molecule.yml configuration file, defines sequence of steps, driver to be used …

Demo


Terminal
"ubi9"
],
"ungrouped": []
}
}
TASK [Assert group existence] ***
ok: [localhost] => {
"changed": false,
"msg": "All assertions passed"
}
PLAY [Converge roles] ***
TASK [Check uname] **
ok: [ubi8]
ok: [ubi9]
TASK [Print some info] **
ok: [ubi8] => {
"changed": false,
"msg": "All assertions passed"
}
ok: [ubi9] => {
"changed": false,
"msg": "All assertions passed"
}
TASK [Include role] ***
TASK [example.web.install_apache : Install apache using commandline] ***********
changed: [ubi9]
changed: [ubi8]
PLAY RECAP **
localhost : ok=3 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
ubi8 : ok=3 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
ubi9 : ok=3 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
CRITICAL Idempotence test failed because of the following tasks:
* => example.web.install_apache : Install apache using commandline
* => example.web.install_apache : Install apache using commandline
[cloud-user@rhel9-molecule extensions]$
0 zsh 1 zsh 2:python3* 3 zsh- 4 zsh 5 zsh
19/11 01:08:27

Red Hat

Red Hat Summit

Connect

Thank you

 linkedin.com/company/red-hat

 facebook.com/redhatinc

 youtube.com/user/RedHatVideos

 twitter.com/RedHat

Red Hat