

Ansible playbooks — это код: проверяем, тестируем, непрерывно интегрируем

Иван Пономарёв, МФТИ / КУРС

ponomarev@corchestra.ru



[@inponomarev](https://twitter.com/inponomarev)

С чем приходится иметь дело:

- Самые простые проекты — классическая «трёхзвенка» (1–2 сервера)
- Самый сложный проект — около 40 серверов (DigitalOcean)
- Terraform + Ansible

Десятки ролей...

```
Ivan@IVAN_MSI ~/choir/roles
$ ls
apt          do_hostname  frontend    mailsender  swap
backoffice  elasticsearch iptables    openresty   test
backups     exim         kibana      parser       timezone
browserpool fastapi      letsencrypt postgres     users
browserpool-sandbox filebeat     locales    s3cmd        zabbix-agent
default-soft fluteauthapi logstash    smashing    zabbix-server
```

*...бэк, фронт, прокси, базы данных,
мониторинг, сбор логов, etc, etc...*

Как всё было:

Проект 1

group_vars/
inventory/
roles/

roleA/

roleB/

roleC/

webservers.yml

database.yml

logs.yml

Проект 2

group_vars/
inventory/
roles/

roleA/

roleB/

roleD/

webservers.yml

database.yml

logs.yml

copy-paste-modify

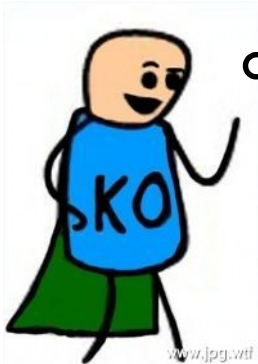
Много кода — знакомые проблемы:

- Страх поломать
 - Код не переиспользуется, а копируется в проекты
 - Нет рефакторинга
- Нет уверенности, что эта куча кода вообще сработает
- Отладка в процессе деплоя

Знакомые проблемы — знакомое решение

— *Автоматическое
тестирование и CI!*

— *Но как?!*



Что мы можем проверить сразу?

Код в гите
вообще
синтаксически
валидный??

1. YAMLLint
2. AnsibleLint
3. `ansible-playbook \`
`--syntax-check`



adrienverge/yamllint

1. синтаксис YAML
2. лишние пробелы
3. переносы строк UNIX-style
4. одинаковость отступов, три дефиса в начале файла...
строже, чем сам Ansible!

yamllint -c **yamllint.yml** .

```
$ yamllint -c yamllint.yml .
```

```
./inventory.yml
```

```
14:81      warning  line too long (83 > 80 characters) (line-length)
```

```
17:81      warning  line too long (90 > 80 characters) (line-length)
```

```
./playbooks/ubuntu-jck.yml
```

```
9:11       warning  truthy value should be true or false (truthy)
```

```
36:26     warning  truthy value should be true or false (truthy)
```

```
47:19     warning  truthy value should be true or false (truthy)
```

```
./playbooks/aix.yml
```

```
9:11      warning  truthy value should be true or false (truthy)
```

настройка правил



willthames/ansible-lint

1. command vs shell module
2. command vs standard modules
(wget, curl, git etc)
3. command/shell idempotence...
строже, чем сам Ansible!
4. легкий фреймворк для создания
своих правил на Python



willthames/ansible-lint

```
ansible-lint \  
--exclude=/var/lib/jenkins/.ansible/roles \  
-v *.yaml
```

Ansible-lint обходит роли,
но стандартные роли
содержат критические ворнинги

ansible-lint -v *.yaml

```
[ANSIBLE0011] All tasks should be named
/home/Ivan/.ansible/roles/jdauphant.nginx/tasks/main.yaml:3
Task/Handler: include_vars {{ item }}

[ANSIBLE0009] Octal file permissions must contain leading zero
/home/Ivan/.ansible/roles/kamaln7.swapfile/tasks/main.yaml:12
Task/Handler: Set swapfile permissions

[ANSIBLE0016] Tasks that run when changed should likely be handlers
/home/Ivan/.ansible/roles/kamaln7.swapfile/tasks/main.yaml:16
Task/Handler: Create swapfile

[ANSIBLE0016] Tasks that run when changed should likely be handlers
/home/Ivan/.ansible/roles/kamaln7.swapfile/tasks/main.yaml:23
Task/Handler: Enable swapfile
```

Syntax check

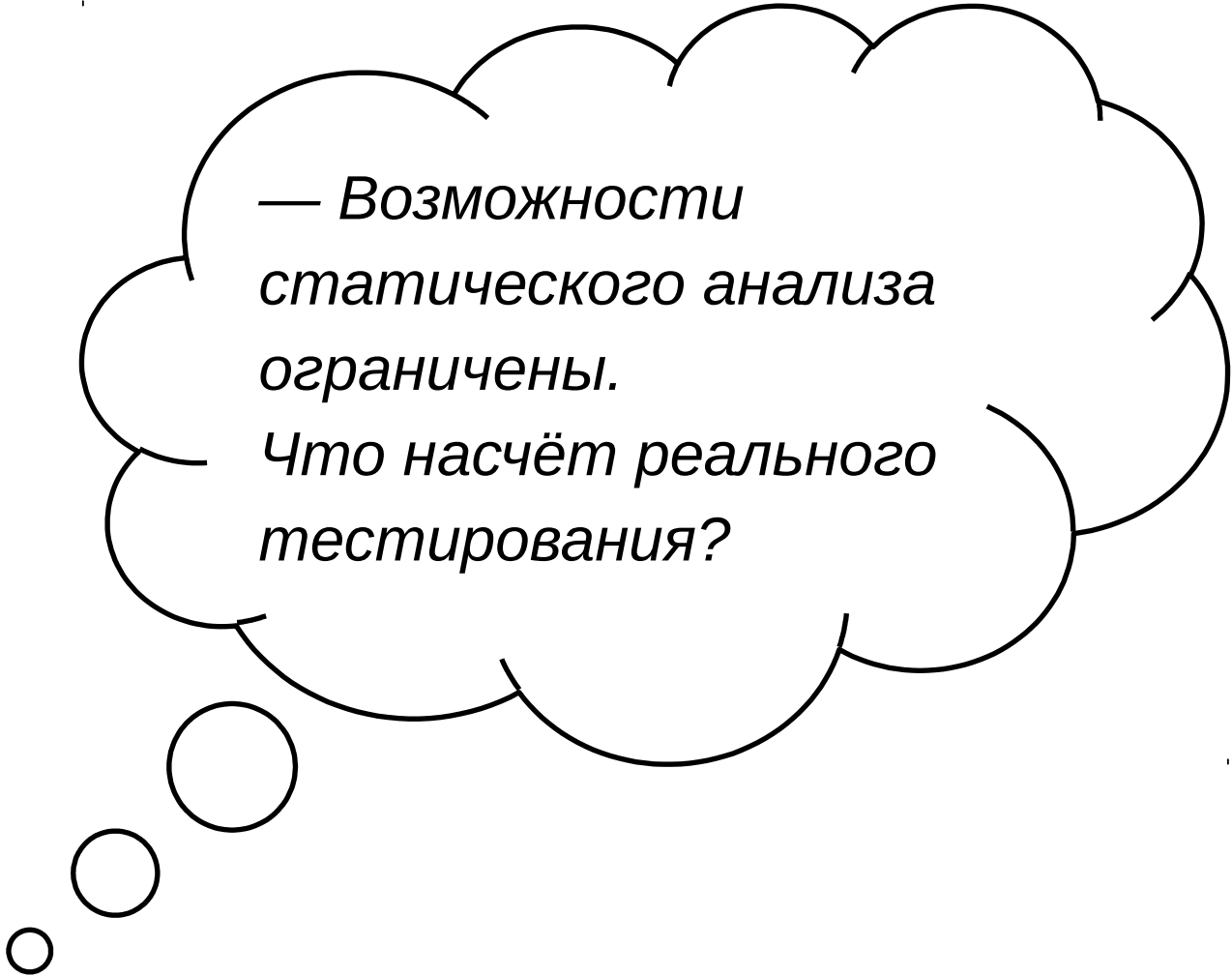
```
ansible-galaxy install -r requirements.yml
```

```
ansible-playbook \  
  playbook.yml --syntax-check
```

*Установите стандартные роли —
иначе проверка синтаксиса свалится
при упоминании неизвестной роли*

Все три инструмента — в CI-скрипт

```
node
{
  stage ('Clone') {
    checkout scm
  }
  stage('YAML lint') {
    sh 'yamllint -c yamllint.yml .'
  }
  stage('Ansible lint') {
    sh 'ansible-galaxy install -r requirements.yml'
    sh 'ansible-lint --exclude=/var/lib/jenkins/.ansible/roles -v *.yaml'
  }
  stage('Ansible syntax check'){
    sh 'ansible-playbook playbook1.yml --syntax-check'
    sh 'ansible-playbook playbook2.yml --syntax-check'
  }
}
```



*— Возможности
статического анализа
ограничены.
Что насчёт реального
тестирования?*

История вопроса

Jeff Geerling

[Blog](#) [Projects](#) [About](#)

Testing Ansible Roles with Travis CI on GitHub

May 23, 2014

This post was originally written in 2014, using a technique that only easily allows testing on Ubuntu 12.04; since then, I've been adapting many of my roles (e.g. [geerlingguy.apache](#)) to use a Docker container-based testing approach, and I've written a new blog post that details the new technique: [How I test Ansible configuration on 7 different OSes with Docker](#).

Since I'm now maintaining [37 roles](#) on Ansible Galaxy, there's no way I can spend as much time reviewing every aspect of every role when doing maintenance, or checking out pull requests to improve the roles. Automated testing using a continuous integration tool like [Travis CI](#) (which is free for public projects and integrated very well with GitHub) allows me to run tests against my Ansible roles with every commit and be more assured nothing broke since the last commit.

Два года спустя...

Jeff Geerling

[Blog](#) [Projects](#) [About](#)

How I test Ansible configuration on 7 different OSes with Docker

October 4, 2016

The following post is an excerpt from chapter 11 in my book [Ansible for DevOps](#). The example used is an [Ansible role that installs Java](#)—since the role is supposed to work across CentOS 6 and 7, Fedora 24, Ubuntu 12.04, 14.04, and 16.04, and Debian 8, I use Docker to run an end-to-end functional test on each of those Linux distributions. See an [example test run in Travis CI](#), and the [Travis file that describes the build](#).

Travis CI



[Blog](#) [Status](#) [Help](#)

Jeff Geerling



geerlingguy / ansible-role-java  build passing

Meet Molecule

 **Watch** 136

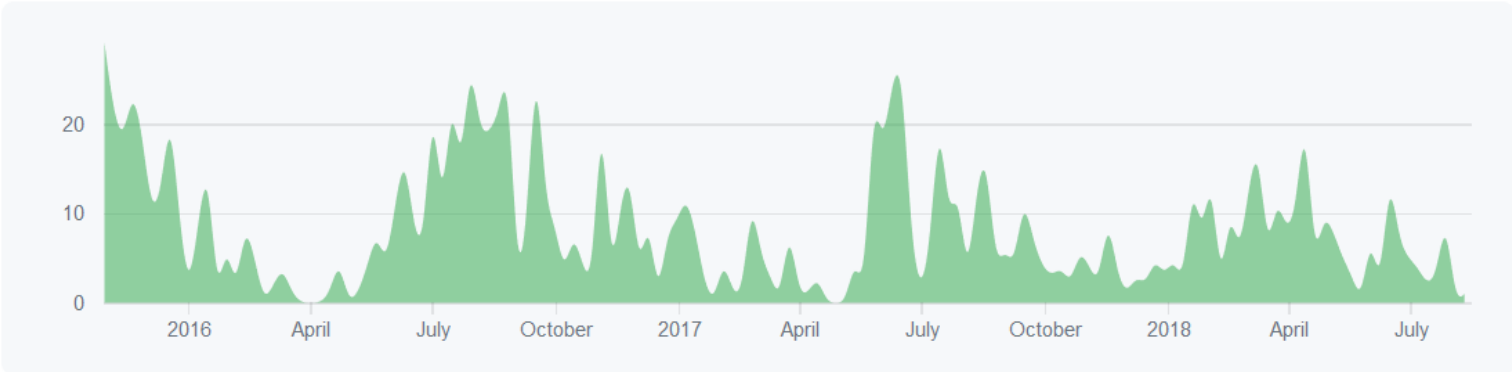
 **Star** 1,390

 **Fork** 211

Nov 15, 2015 – Sep 1, 2018

Contributions: **Commits** ▾

Contributions to master, excluding merge commits



Зависимости Molecule от других проектов

 [adrienverge/yamllint](#)

 [willthames/ansible-lint](#)

 [rusqa/flake8](#)

 [philpep/testinfra](#)

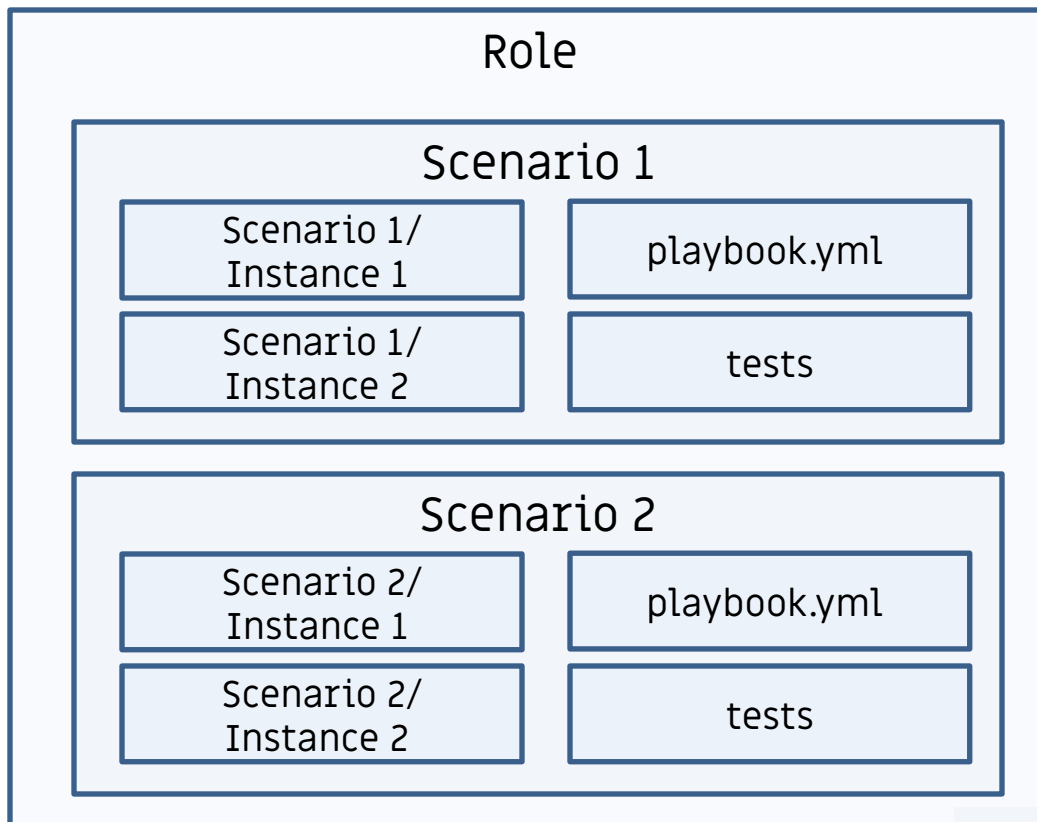
...

УСТАНОВКА

Prerequisites (Ubuntu): gcc, python-dev

- `pip install ansible`
- `pip install molecule`
- `pip install docker-py`

Структура проекта molecule



Инициализация

Новая роль:

```
molecule init role -r newrole
```

```
root@ubuntu-s-1vcpu-1gb-1on1-01:~/tmp/newrole# ls
defaults handlers meta molecule README.md tasks vars
root@ubuntu-s-1vcpu-1gb-1on1-01:~/tmp/newrole#
root@ubuntu-s-1vcpu-1gb-1on1-01:~/tmp/newrole# cd molecule/
root@ubuntu-s-1vcpu-1gb-1on1-01:~/tmp/newrole/molecule# ls
default
root@ubuntu-s-1vcpu-1gb-1on1-01:~/tmp/newrole/molecule# |
```

Инициализация

Существующая роль:

```
molecule init scenario -r <your_role_name>
```

(Ключом --scenario-name можно задать имя сценария, по умолчанию — default)

Запуск

molecule test

```
root@m:~/fluteansible# molecule test
--> Validating schema /root/fluteansible/molecule/default/molecule.yml.
Validation completed successfully.
--> Test matrix
```

```
TASK [wait for instance(s) deletion to complete] *****
***
  failed: [localhost] (item=None) => {"attempts": 1, "censored": "the output
as been hidden due to the fact that 'no_log: true' was specified for this res
", "changed": false}

PLAY RECAP *****
***
localhost                : ok=1    changed=1    unreachable=0    failed=0
```


Запуск

```
molecule --debug test
```

```
root@m:~/fluteansible# molecule test
--> Validating schema /root/fluteansible/molecule/default/molecule.yml.
Validation completed successfully.
--> Test matrix
```

```
      "msg": "Failed to import docker-py - No module named docker. Try `pip
install docker-py`"
    }
}
```

```
PLAY RECAP *****
```

“Test matrix”

```
--> Test matrix
```

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

Настройка instances

molecule/default/molecule.yml

driver:

name: docker

platforms:

- name: ubuntuinstance

image: solita/ubuntu-systemd:latest

command: /sbin/init

privileged: True

volumes:

-

"/sys/fs/cgroup:/sys/fs/cgroup:rw"

- name: centosinstance

image: solita/centos-systemd:latest

command: /sbin/init

privileged: True

volumes:

- "/sys/fs/cgroup:/sys/fs/cgroup:rw" .

Кроме docker, есть драйверы

- Azure (via Ansible's `azure_module`)
- EC2 (via Ansible's `ec2_module`)
- GCE (via Ansible's `gce_module`)
- Delegated (самостоятельное определение `create/destroy/ssh params`)
- Vagrant
- ...and more

Подключение зависимостей

molecule/default/molecule.yml

```
dependency:  
  name: galaxy  
  options:  
    role-file: requirements.yml
```

requirements.yml

```
---  
- src: ansiblebit.oracle-java  
  version: "5.14.14"
```

```
--> Action: 'dependency'  
- downloading role 'oracle-java', owned by ansiblebit  
- downloading role from https://github.com/ansiblebit/oracle-java/archive/5.14.14.tar.gz  
- extracting ansiblebit.oracle-java to /cygdrive/d/ideospace/fluteansible/molecule/default/.molecule/roles/ansiblebit.oracle-java  
- ansiblebit.oracle-java (5.14.14) was installed successfully
```

Этап статического анализа (lint)

- `yamllint`
- `ansible-lint`
- `ansible-playbook --syntax-check`

Converge

molecule/default/playbook.yml

```
---
```

```
- name: Converge
  hosts: all
  roles:
    - role: ansiblebit.oracle-java
    - role: fluteansible
  tasks:
    ...
```

Подключение к тестовой ноде для отладки результатов выполнения

```
molecule test --destroy=never  
docker exec -it instance /bin/bash
```


Проверка идемпотентности

Раньше (из статьи Jeff Geerling):

```
ansible-playbook -i tests/inventory tests/test.yml
--connection=local --sudo
| grep -q 'changed=0.*failed=0'
&& (echo 'Idempotence test: pass' && exit 0)
|| (echo 'Idempotence test: fail' && exit 1)
```

Проверка идемпотентности

Теперь (в Molecule):

```
ansible-playbook --diff playbook.yml
```

Инфраструктурные тесты

Molecule поддерживает:

- Testinfra (Python, default)
- Serverspec (Ruby)
- Goss (written in Go, tests in YAML)

Testinfra bootstrap

```
import os

import testinfra.utils.ansible_runner

testinfra_hosts =
    testinfra.utils.ansible_runner.AnsibleRunner(
        os.environ['MOLECULE_INVENTORY_FILE']).get_hosts('all')
```

Результат запуска команды

```
def test_jython_installed(host):  
    cmd = host.run('jython --version')  
    assert cmd.rc == 0  
    assert cmd.stderr.find(u'Jython 2') > -1
```

«УМНЫЙ» assert

```
> assert cmd.stderr.find(u'Jython 3') > -1
E     AssertionError: assert -1 > -1
E     + where -1 = <built-in method find of unicode object at 0x7f69364e5a20>
>('Jython 3')
E     + where <built-in method find of unicode object at 0x7f69364e5a20> =
'Jython 2.7.1b3'.find
E     + where 'Jython 2.7.1b3' = CommandResult(command='jython --version
', exit_status=0, stdout=u'', stderr=u'Jython 2.7.1b3').stderr

tests/test_default.py:17: AssertionError
===== 1 failed, 3 passed in 11.49 seconds =====
```

Быстрый перезапуск тестов

```
molecule test --destroy=never
```

```
<дописываем тест...>
```

```
molecule verify
```

```
<дописываем тест...>
```

```
molecule verify
```

'Keep the bar green to keep your infrastructure clean'

```
pragmst@es5hmta:~$ python3 -m unittest discover
collected 4 items

tests/test_default.py .... [100%]

===== 4 passed in 12.04 seconds =====
Verifier completed successfully.
```


Веб-сервисы

(Предварительно установив curl в playbook.yml)

```
def test_service_greeting(host):  
    cmd = 'curl -o -I -L -s -w "%{http_code}\n"  
http://localhost:8080/service'  
    assert host.check_output(cmd) == '200'  
    cmd = "curl -L 'http://localhost:8080/service'"  
    assert host.check_output(cmd).find(u'Hello!') > -1
```

Процессы

```
def test_jsvc_process(host):  
    procs = host.process.filter(comm="jsvc")  
    assert len(procs) > 0  
    for proc in procs:  
        assert proc.user == 'flute3'  
        assert proc.args.find('-Xmx1024M') > -1
```

Сервисы

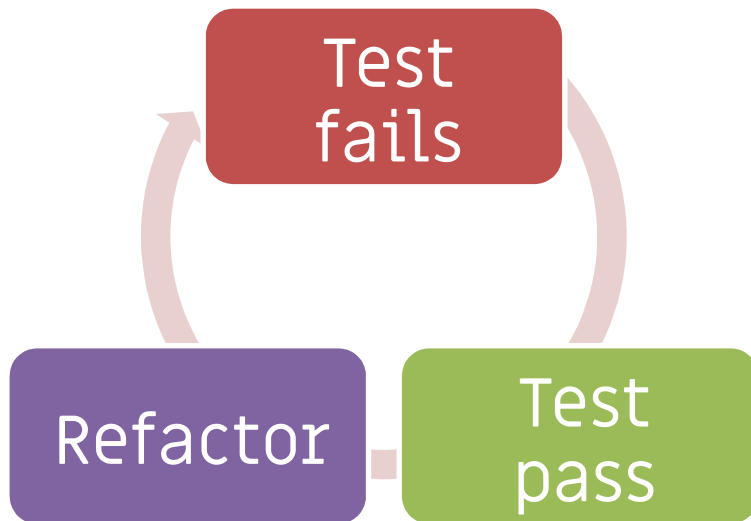
```
def test_service_is_running(host):  
    assert host.service('flute').is_running
```

Файлы и их содержимое

```
def test_log_files(host):  
    stdout = host.file('/var/log/flute/std.out')  
    stderr = host.file('/var/log/flute/std.err')  
    assert stdout.exists  
    assert stderr.exists  
    assert stderr.contains('Flute started')  
    assert stdout.contains('Flute started. 0  
taskSources are being processed')
```

— TDD for
Ansible?

— Ну, если
хотите — да!



— Но и в самом
Ansible есть
модуль **assert!**

- name: call for jython version
command: jython --version
register: cmd_result

ERROR: Idempotence test failed

- name: check jython
assert:
that: "'Jython 2' in cmd_result.stderr"



Проверки в хэндлерах

tasks/main.yml

- name: download jython
...
notify: "validate jython"
- name: install jython
...
notify: "validate jython"

handlers/main.yml

- name: call for jython version
command: jython --version
register: cmd_result
listen: "validate jython"
- name: check jython
assert:
 that: "'jython 2' in cmd_result.stderr"
listen: "validate jython"

Ещё можно проверить в хэндлерах:

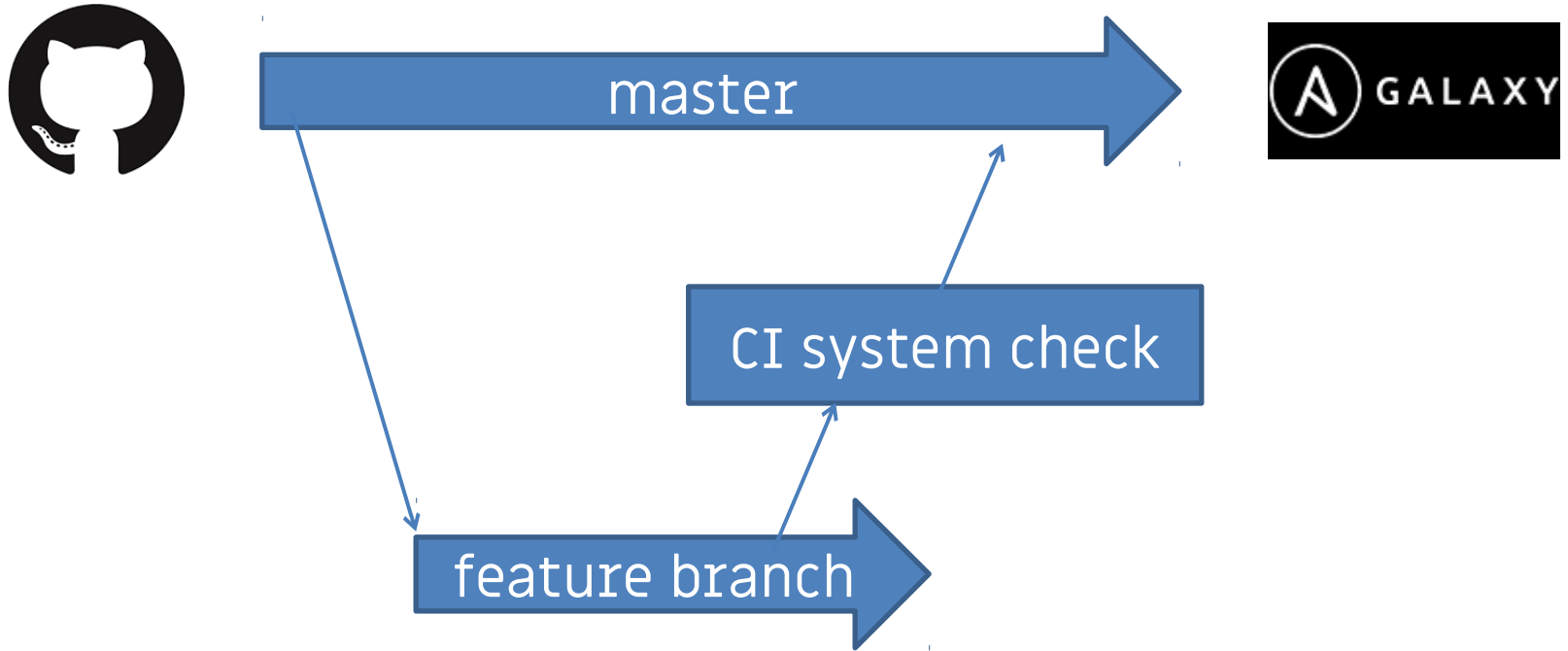
файлы:

- stat:
 - path: /path/to/something
 - register: p
- assert:
 - that:
 - "p.stat.exists and
p.stat.isdir"

веб-сервисы:

- uri:
 - url: http://www.example.com
 - return_content: yes
 - register: webpage
- assert:
 - that:
 - "'AWESOME' in
webpage.content"

Role development process



CI: Jenkins Multibranch Pipeline

```
node {  
  stage ("Get Latest Code") {  
    checkout scm  
  }  
  
  stage ("Molecule test") {  
    sh 'mkdir -p molecule/default/roles'  
    sh 'ln -sf `pwd` molecule/default/roles/fluteansible'  
    sh 'molecule test'  
  }  
}
```

Разделение по стадиям

```
stage ("Executing molecule lint") {
  sh 'molecule lint'
}
stage ("Executing molecule create") {
  sh 'molecule create'
}
stage ("Executing molecule converge") {
  sh 'molecule converge'
}
stage ("Executing molecule idempotence") {
  sh 'molecule idempotence'
}
stage ("Executing molecule verify") {
  sh 'molecule verify'
}
```

(код из статьи **Werner Dijkerman**, "Continuous deployment of Ansible Roles")

Разделение по стадиям



(Image from: <https://werner-dijkerman.nl/2017/09/17/continuous-deployment-of-ansible-roles/>)

Travis-CI

```
language: python
python: "2.7"
sudo: required
```

```
services:
- docker
```

```
before_install:
- sudo apt-get update -qq
```

```
install:
- pip install ansible==2.5.0
- pip install molecule
- pip install docker-py
```

```
script:
- molecule test
```

```
notifications:
webhooks: https://
galaxy.ansible.com/api/v1/notifications/
```

nginx 114179

nginx installation for Linux, FreeBSD and OpenBSD.

| | |
|-------------|---|
| Type | Ansible |
| Author | geerlingguy |
| OS | Debian, Enterprise_Linux, FreeBSD, O... |
| Clouds | NA |
| Tags | balancer, development, load, nginx, pr... |
| Last Commit | 8 days ago |

build passing Watch 27 Star 355

nginx 44689

Ansible role to install and manage nginx configuration

| | |
|-------------|---|
| Type | Ansible |
| Author | jdauphant |
| OS | Debian, Enterprise_Linux, Fedora, Free... |
| Clouds | NA |
| Tags | web |
| Last Commit | 10 days ago |

build passing Watch 37 Star 544

Nginx 11289

Install and configure Nginx and Phusion Passenger along with any number of server blocks (aka vhosts).

| | |
|-------------|---|
| Type | Ansible |
| Author | bbatsche |
| OS | Ubuntu |
| Clouds | NA |
| Tags | hhvm, nginx, node, passenger, php, p... |
| Last Commit | 2 months ago |

build failing Watch 1 Star 1

nginx 4310

Install and manage nginx webserver

| | |
|-------------|------------------------|
| Type | Ansible |
| Author | debops |
| OS | Debian, Ubuntu |
| Clouds | NA |
| Tags | nginx, web, webserver |
| Last Commit | 6 months ago |

build passing Watch 15 Star 42

Было...

Проект 1

group_vars/
inventory/
roles/

roleA/

roleB/

roleC/

webservers.yml

database.yml

logs.yml

Проект 2

group_vars/
inventory/
roles/

roleA/

roleB/

roleD/

webservers.yml

database.yml

logs.yml

copy-paste-modify

...СТАЛО:

Ansible Galaxy

roleA/ roleB/

Molecule

Проект 1

group_vars/
inventory/
roles/
 roleC/
webservers.yml
database.yml
logs.yml

Проект 2

group_vars/
inventory/
roles/
 roleD/
webservers.yml
database.yml
logs.yml

Linting

Можно ещё что-то улучшить?

Ansible Galaxy

roleA/ roleB/

Проект 1

group_vars/
inventory/
roles/

roleC/

webservers.yml
database.yml
logs.yml

Проект 2

group_vars/
inventory/
roles/

roleD/

webservers.yml
database.yml
logs.yml

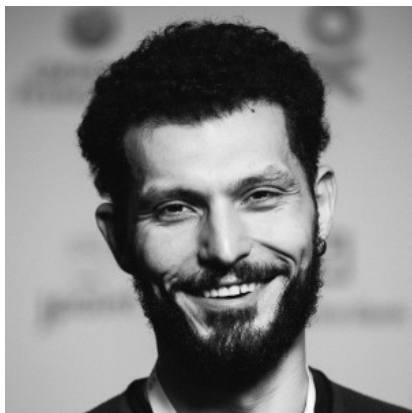
Как быть с конфигурацией?

- Плохие новости:
 - Molecule — только для ролей.
 - Проверить развёртывание на прод можно только развернув на прод.))
- Хорошая новость:
 - Мы можем проверить проект, не запуская его!



Андрей Сатарин

«Как проверить систему, не запуская её»
Heisenbug 2017, Москва



Руслан Черемин

«Тестирование конфигурации
для Java-разработчиков: практический опыт»
Heisenbug 2018, Санкт-Петербург
JUG.MSK, июнь 2018, Москва

Тесты конфигурации

- Нужные переменные вообще есть.
- Формат значений переменных:
 - порты — числа в допустимом диапазоне,
 - хосты — доменные имена или IP,
 - URL'ы — валидные и т. п.
- Паролей нет в явном виде.
- Порты сервисов, поднимаемых в пределах хоста, уникальны.
- И более специфические вещи.

Проверяем валидность портов

```
@pytest.mark.parametrize("k,v", port_var_values(BASEDIR))
def test_port(k, v):
    port = int(v)
    assert port > 0
    assert port < 32000
```

Обходим нужные нам переменные...

```
def port_var_values(path):  
    key_pattern = re.compile('port$')  
    for (k, v) in var_values(path):  
        if key_pattern.search(k):  
            yield (k, v)
```

Проверяем уникальность портов

```
@pytest.mark.parametrize("path", [BASEDIR])
def test_port_uniqueness(path):
    ports = set()
    for (k, v) in port_var_values(path):
        port = int(v)
        assert not (port in ports)
        ports.add(port)
```


'Keep the bar green to keep the configuration clean'

```
$ pytest
===== test session starts =====
platform cygwin -- Python 2.7.14, pytest-3.6.1, py-1.5.3, pluggy-0.6.0
rootdir: /cygdrive/d/ideospace/untitled5/test, inifile:
plugins: testinfra-1.7.1
collected 101 items

test_config.py ..... [ 56%]
..... [100%]

===== 101 passed in 1.79 seconds =====
```

ЛОВИМ «УТЕКАЮЩИЕ» ПАРОЛИ

```
def password_var_values(path):  
    key_pattern = re.compile('p(ass(word)?|wd)$')  
    for (k, v) in var_values(path):  
        if key_pattern.search(k):  
            yield (k, v)  
  
@pytest.mark.parametrize("k,v", password_var_values(BASEDIR))  
def test_password(k, v):  
    var_pattern = re.compile('\\{\\{([^}]|\\}[^}]*)+\\}\\}')  
    print '%s: %s' % (k, v)  
    assert var_pattern.search(v)
```

Ай-яй-яй!

```
test_password[mysql_root_password-12345]
```

```
k = 'mysql_root_password', v = '12345'
```

```
@pytest.mark.parametrize("k,v", all_vars_values(BASEDIR))
```

```
def test_password(k, v):
```

```
    var_pattern = re.compile('\{\{([^\}|\}[^})]+\}\}')
```

```
    if is_password(k):
```

```
        print '%s: %s' % (k, v)
```

```
        assert var_pattern.search(v)
```

```
> E AssertionError: assert None
```

```
E + where None = <built-in method search of _sre.SRE_Pattern object at 0x6fffe92c450>('12345')
```

```
E + where <built-in method search of _sre.SRE_Pattern object at 0x6fffe92c450> = <_sre.SRE_Pattern object at 0x6fffe92c450>.search
```

```
test_hello.py:52: AssertionError
```

```
----- Captured stdout call -----
```

```
mysql_root_password: 12345
```

Выводы

Тестируйте ваш Ansible!

- YAMLLint + AnsibleLint + syntax check **прямо сегодня!**
- Проверяйте роли на Molecule
- Вставляйте проверки в хэндлеры
- Тестируйте конфигурацию

Molecule is 'must have' при разработке ролей

- Создайте тесты на ваши Ansible-роли прямо сегодня — это просто
- Лень разбираться?
Без тестов, проверка converge и idempotence
- Совсем лень разбираться? lint /syntax

**Назвался кодом —
полезай в СИ!**

CI/CD для Ansible-ролей

- «Штатный» набор инструментов — GitHub+Travis+Galaxy, это если роли в OpenSource
- Jenkins Multibranch тоже работает отлично

**Тестировать Ansible-код —
просто и приятно!**

Задавайте вопросы :-)

ponomarev@corchestra.ru

 [@inponomarev](https://twitter.com/inponomarev)

ССЫЛКИ

- **Иван Пономарёв** Тестирование и непрерывная интеграция для Ansible-ролей при помощи Molecule и Jenkins <https://habr.com/post/351974/>
- **Jeff Geerling** Testing Ansible Roles with Travis CI on GitHub <https://www.jeffgeerling.com/blog/testing-ansible-roles-travis-ci-github>
- **Werner Dijkerman** Continuous deployment of Ansible Roles <https://werner-dijkerman.nl/2017/09/17/continuous-deployment-of-ansible-roles/>