Локальная разработка и тестирование Ansible ролей и плейбуков

План

- Тестирование инфраструктурного кода
- Инструменты и примеры
- Практики из разработки

Зачем тестировать?

- Улучшаем и контролируем качество
- Получаем обратную связь
- Экономим время за счет автоматизации
- Тесты как документация

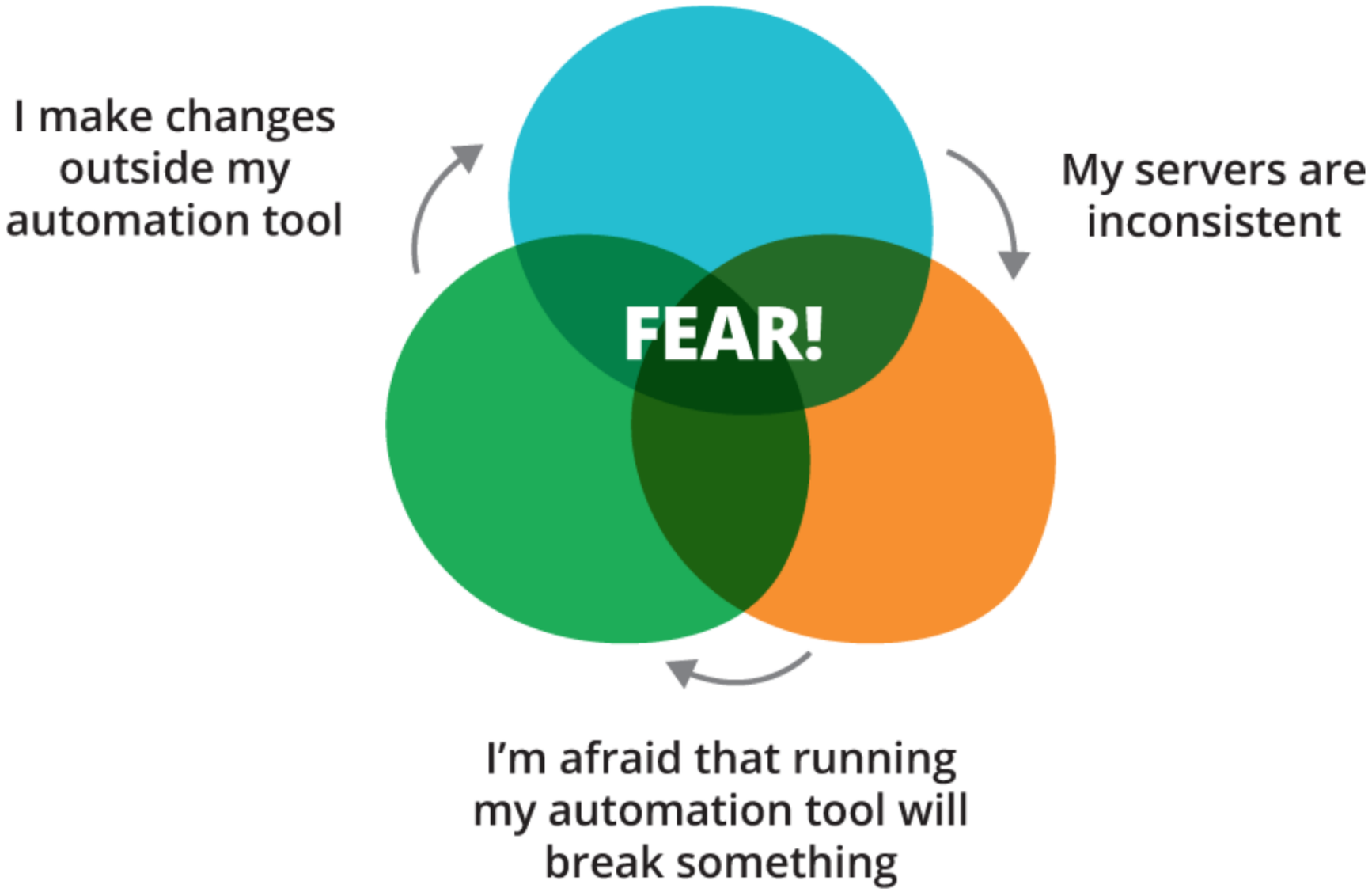
Когда актуально?

- Изменения
- Коллектив
- Сложная логика
- Чужой код

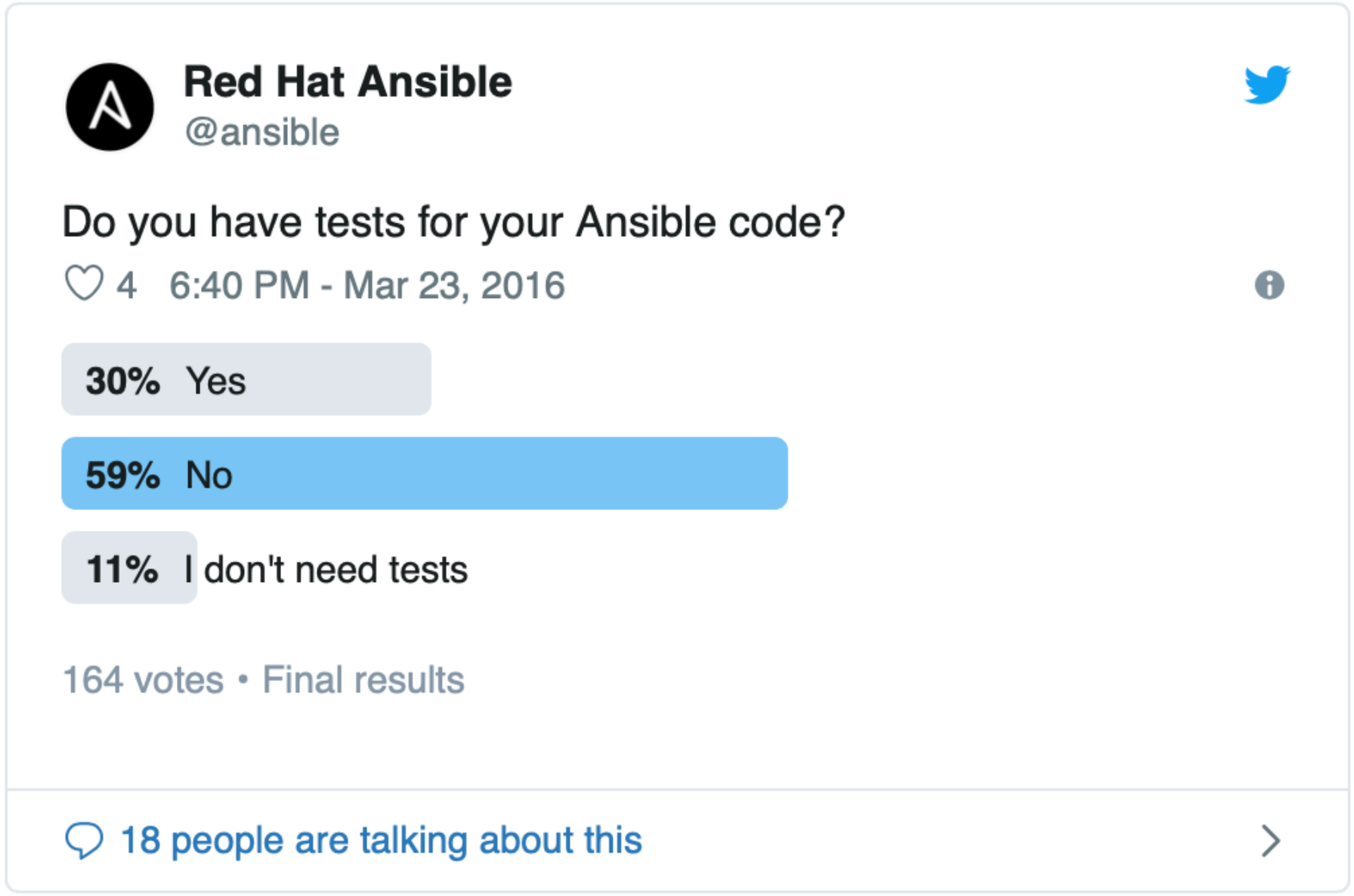
Изменения

- Внутренние изменения:
  - Релизы, инциденты, смена стека, …
- Внешние изменения:
  - Новая версия роли
  - Новая версия пакета
  - Новая версия Ansible
  - Новая версия ОС

Страх автоматизации



Ansible и тесты



Из официальной документации

- Ansible: fail-fast, order-based, declarative, …
- Модули и роли идемпотентны
- Используйте dry run (-check)
- Используйте модули uri, script, assert, stat
- Используйте Rolling update и pre/post таски
- Дополнительные тестовые инструменты не нужны

Ansible в роли догоняющего

- Chef и Puppet ушли вперед:
  - Начали раньше развивать
  - Больше охват рынка
  - Есть успешные примеры
  - Есть выбор из инструментов тестирования
  - Появилась официальная поддержка (DK)

Что можем протестировать?

- Стиль языка и инфраструктурного кода
- Валидацию кода
- Функционал:
  - Интеграцию
  - Результат работы кода

Что можем протестировать?

- **Стиль языка и инфраструктурного кода**
- Валидацию кода
- Функционал:
  - Интеграцию
  - Результат работы кода

Стиль языка и кода

- Набор правил
- На основе Style Guides и Best Practices
- Линтеры для языка Python:
  - pep8
- Линтеры для инфраструктурного кода Ansible:
  - yamllint, ansible-lint

ansible-lint

- CLI утилита
- Есть интеграция в IDE и Git
- ~ 20 правил
- Конфигурация в .ansible-lint
- Используется в Ansible Galaxy

ansible-lint

```
$ ansible-lint db.yml

    57:42     error    no new line character at the end of file  (new-line-at-end-of-file)

  /Users/test/mol/ansible/roles/db/tasks/configure.yml
    1:1       warning  missing document start "---"  (document-start)

  /Users/test/mol/ansible/roles/db/tasks/install.yml
    3:1       warning  missing document start "---"  (document-start)

  /Users/test/mol/ansible/roles/db/tests/test.yml
    5:9       error    no new line character at the end of file  (new-line-at-end-of-file)

  /Users/test/mol/ansible/roles/db/vars/main.yml
    2:19      error    no new line character at the end of file  (new-line-at-end-of-file)
```

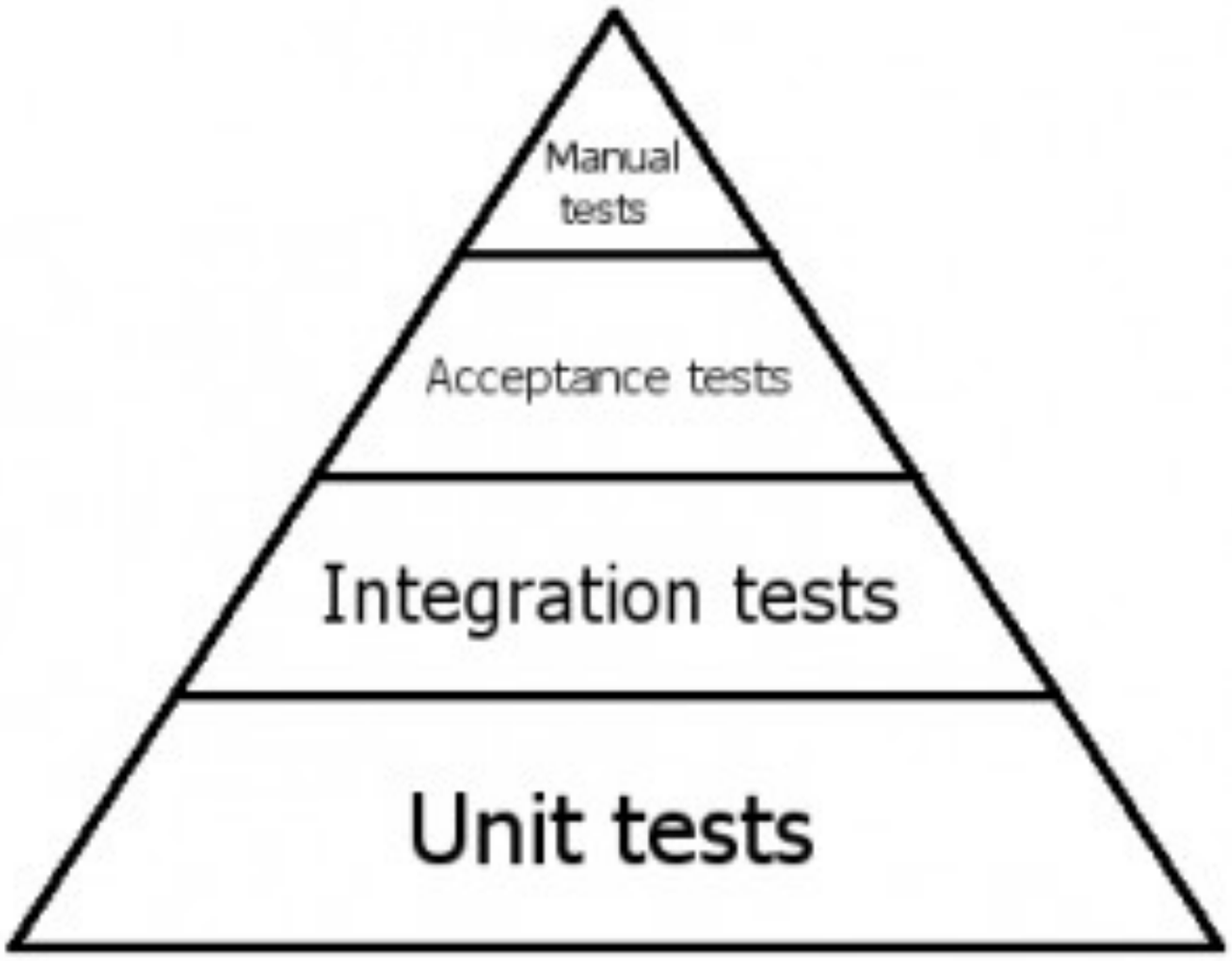
Что можем протестировать?

- Стиль языка и инфраструктурного кода
- **Валидацию кода**
- Функционал:
  - Интеграцию
  - Результат работы кода

Что можем протестировать?

- Стиль языка и инфраструктурного кода
- Валидацию кода
- **Функционал:**
  - **Интеграцию**
  - **Результат работы кода**

Пирамида тестирования



Функционал

- Модульное тестирование (Unit testing)
- Интеграционное тестирование (Integration testing)
- Приемочное тестирование (Acceptance testing)

Функционал

- Модульное тестирование (Unit testing)
- **Интеграционное тестирование (Integration testing)**
- **Приемочное тестирование (Acceptance testing)**

Инфраструктурные тестовые фреймворки

- Проверяют состояние инфраструктуры - локальных или удаленных окружений

- Являются расширением обычных тестовых фреймворков*:

  - Используют DSL (Domain Specific Language)

  - Добавляют ресурсы или модули (Host, Package, Process, Service, …)

  - Поддерживают разные форматы вывода

Инфраструктурные тестовые фреймворки

- Serverspec
- InSpec
- Testinfra
- Goss

Сравнение фреймворков

name	testinfra	serverspec	inspec	Goss
link	testinfra.readthedocs.io	serverspec.org	www.inspec.io	goss.rocks
source	philpep/testinfra	mizzy/serverspec	chef/inspec	aelsabbahy/goss
language	python	ruby	ruby	go
Watch	93	145	165	67
Star	997	2105	1167	2170
Fork	138	361	330	156
license	Apache 2.0	MIT	Apache 2.0	Apache 2.0
commits	380	1854	4609	309
releases	35	282	346	47
contributors	43	110	159	31

Testinfra

- Написан на Python
- Является расширением над Pytest
- ~ 20 готовых модулей
- В основном используется для Salt и Ansible

Пример теста

db/molecule/default/tests/test_default.py

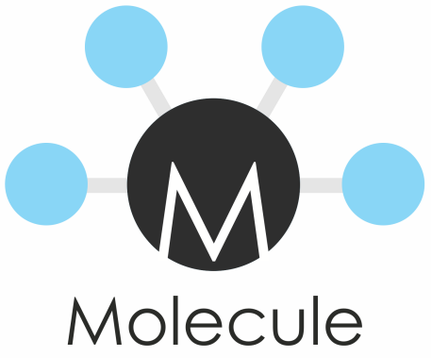
```
def test_mongo_running_and_enabled(host):
    mongo = host.service("mongod")
    assert mongo.is_running
    assert mongo.is_enabled

def test_config_file(File):
    config_file = File(‘/etc/mongod.conf')
    assert config_file.is_file
    assert config_file.contains('bindIp: 0.0.0.0')

def test_socket_listening(Socket):
    socket = Socket('tcp://0.0.0.0:27017')
    assert socket.is_listening
```

Ручной процесс

- Прогнать линтеры
- Запустить виртуальную машину
- Запустить ansible
- Посмотреть diff изменений
- Запустить инфраструктурные тесты
- Проверить результат
- Остановить виртуальную машину

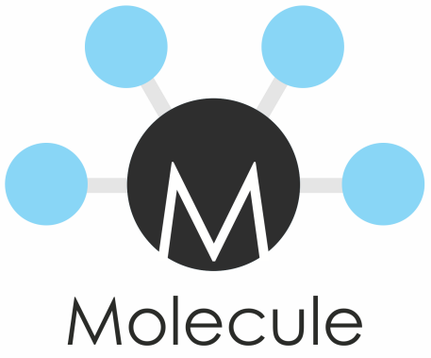
Автоматизируем

- Инструменты:
  - Vagrant
  - Test Kitchen
  - Molecule (Ansible)
  - Beaker (Puppet)



Автоматизируем

- Инструменты:
  - **Vagrant**
  - **Test Kitchen**
  - **Molecule (Ansible)**
  - Beaker (Puppet)



Vagrant

- Инструмент для автоматизации разработки:
  - Создание, настройка и удаление локальных окружений
  - Providers (VirtualBox, Hyper-W, VMware, Docker, …)
  - Provisioners (Ansible, Chef, Puppet, Salt, Shell, …)
  - Плагины

Vagrant

- Появился первым, еще в 2010 году
- Простой workflow:
  - up, provision, destroy
- Универсальное решение:
  - Не учитывается специфику инструментов
- Vagrantfile:
  - Ruby DSL + Ruby
  - Подходит для простых конфигураций

Пример Vagrantfile

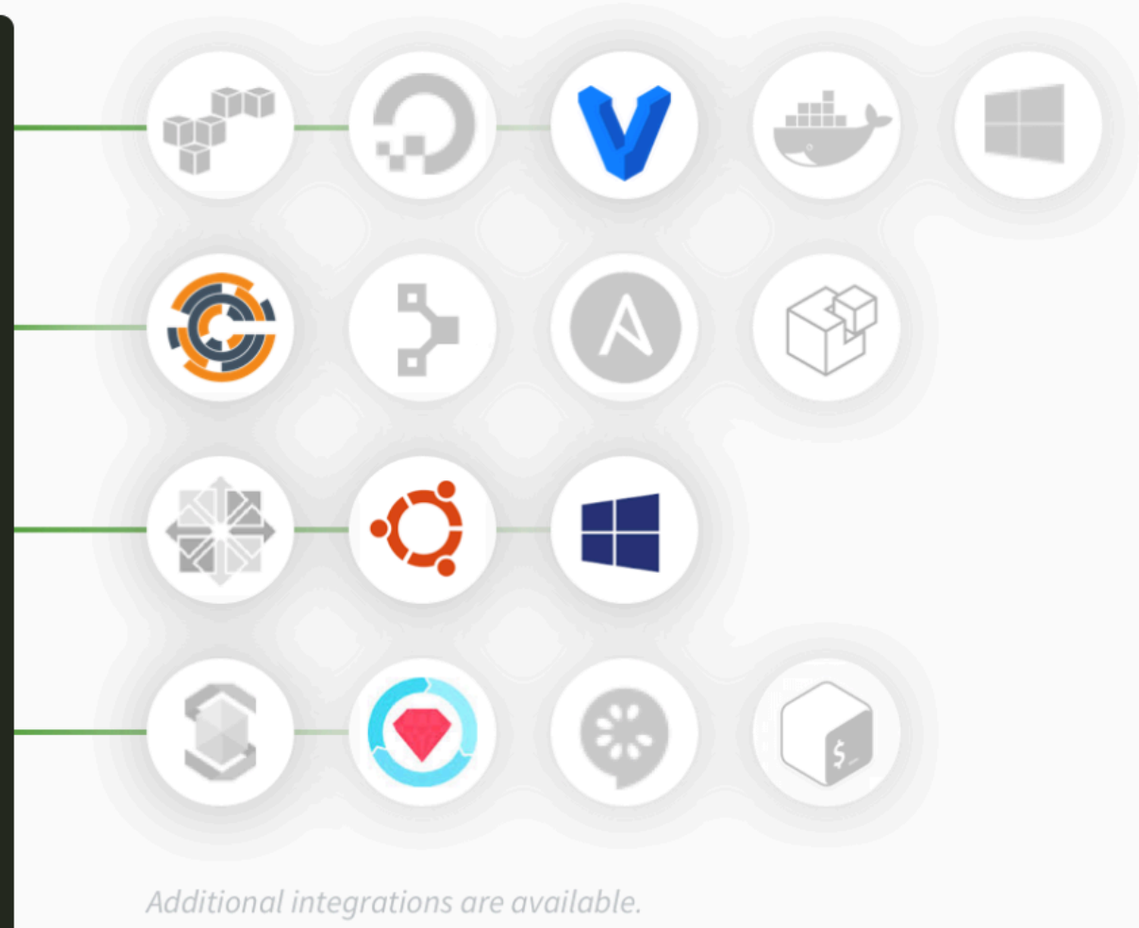
```
Vagrant.configure("2") do |config|

  config.vm.define "ubuntu16" do |vm|
    vm.vm.box = "ubuntu/xenial64"
  end

  config.vm.provider :virtualbox do |vb|
    vb.memory = 512
  end

  config.vm.provision "ansible" do |ansible|
      ansible.playbook = "db.yml"
  end
end
```

Test Kitchen

- Все тоже самое что и в Vagrant (Drivers, Provisioners)
- Конфигурация в YAML
- Локальная CI:
  - Фокус на тестировании
  - Тестовые фреймворки (Serverspec, InSpec, Testinfra, Goss, …)
  - Наборы тестов
- Универсальное решение (Chef, Puppet, Ansible, Salt)
- Написан на Ruby

Test Kitchen

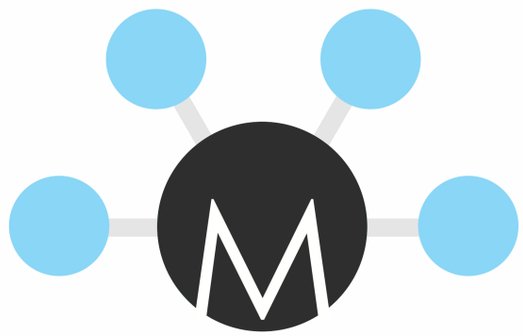
```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-14.04
  - name: windows-2012r2

suites:
  - name: client
    run_list:
      - recipe[postgresql::client]
  - name: server
    run_list:
      - recipe[postgresql::server]
```



Additional integrations are available.

Molecule

- Аналог Test Kitchen
- Написан на Python



Molecule

- Конфигурация в YAML
- Официальный инструмент с сентября 2018
- Интеграция с экосистемой Ansible (workflow и инструменты)

molecule --help

```
$ molecule --help
Usage: molecule [OPTIONS] COMMAND [ARGS]...

Commands:
  check        Use the provisioner to perform a Dry-Run...
  converge     Use the provisioner to configure instances...
  create       Use the provisioner to start the instances.
  dependency   Manage the role's dependencies.
  destroy      Use the provisioner to destroy the instances.
  idempotence  Use the provisioner to configure the...
  init         Initialize a new role or scenario.
  lint         Lint the role.
  list         Lists status of instances.
  login        Log in to one instance.
  matrix       List matrix of steps used to test instances.
  prepare      Use the provisioner to prepare the instances...
  side-effect  Use the provisioner to perform side-effects...
  syntax       Use the provisioner to syntax check the role.
  test         Test (lint, destroy, dependency, syntax,...
  verify       Run automated tests against instances.
```

molecule init

- Позволяет создать заготовку роли с тестами на основе cookiecutter шаблона
- Есть аналог ansible-galaxy init

```
# пример
$ molecule init role --role-name db --driver-name vagrant

# для готовой роли
$ molecule init scenario --scenario-name default -r db -d vagrant
```

molecule.yml

```
---
driver:
  name: vagrant
  provider:
    name: virtualbox
platforms:
  - name: instance
    box: ubuntu/xenial64
provisioner:
  name: ansible
  lint:
    name: ansible-lint
verifier:
  name: testinfra
```

Пример

$ molecule create

--> Test matrix

```
└─── default
    └─── create
--> Scenario: 'default'
--> Action: 'create'

    PLAY [Create] ******************************************************************

    TASK [Create molecule instance(s)] *********************************************
    changed: [localhost] => (item={'box': u'ubuntu/xenial64', 'name': u'instance'})

    TASK [Populate instance config dict] *******************************************
    ok: [localhost] => (item={u'IdentityFile': u'/Users/test/mol/ansible/roles/db/molecule/default/.molecule/.vagrant/
machines/instance/virtualbox/private_key', '_ansible_parsed': True, '_ansible_item_result': True, '_ansible_no_log':
False, 'item': {'box': u'ubuntu/xenial64', 'name': u'instance'}, u'LogLevel': u'FATAL', u'HostName': u'127.0.0.1',
u'Host': u'instance', u'PasswordAuthentication': u'no', u'IdentitiesOnly': u'yes', u'UserKnownHostsFile': u'/dev/null',
u'User': u'ubuntu', u'changed': True, u'invocation': {u'module_args': {u'platform_box_url': None, u'provider_cpus': 2,
u'provider_raw_config_args': None, u'platform_box': u'ubuntu/xenial64', u'platform_box_version': None,
u'instance_interfaces': [], u'instance_name': u'instance', u'state': u'up', u'provider_memory': 512,
u'instance_raw_config_args': None, u'provider_options': {}, u'provider_name': u'virtualbox', u'force_stop': False}},
u'StrictHostKeyChecking': u'no', u'Port': u'2222'})
```

Пример

```
$ molecule converge

--> Test matrix

└── default
    ├── create
    └── converge
--> Scenario: 'default'
--> Action: 'create'
Skipping, instances already created.
--> Scenario: 'default'
--> Action: 'converge'

    TASK [db : Configure service supervisor] ***************************************
    changed: [instance]

    TASK [db : Show info about the env this host belongs to] ***********************
    ok: [instance] => {
        "msg": "This host is in local environment!!!"
    }

    TASK [db : Change mongo config file] *******************************************
    changed: [instance]

    RUNNING HANDLER [db : restart mongod] ******************************************
    changed: [instance]

    PLAY RECAP *********************************************************************
    instance                   : ok=9    changed=6    unreachable=0    failed=0
```

Пример

Первый тест прошел, два зафейлилось

```
$ molecule verify
collected 3 items

    tests/test_default.py .FF

    ================================ FAILURES =================================
      File = <class 'testinfra.modules.base.GNUFile'>

        def test_config_file(File):
            config_file = File('/etc/mongod.conf')
    >       assert config_file.contains('bindIp: 0.0.0.0')
    E       AssertionError: assert False
    E        +  where False = <bound method GNUFile.contains of <file /etc/mongod.conf>>('bindIp:
0.0.0.0')
    E        +    where <bound method GNUFile.contains of <file /etc/mongod.conf>> = <file /etc/
mongod.conf>.contains
     tests/test_default.py:15: AssertionError
    _____________ test_socket_listening[ansible://instance] ___________________

    Socket = <class 'testinfra.modules.base.LinuxSocket'>

        def test_socket_listening(Socket):
            socket = Socket('tcp://0.0.0.0:27017')
    >       assert socket.is_listening
    E       assert False
    E        +  where False = <socket tcp://0.0.0.0:27017>.is_listening
    tests/test_default.py:20: AssertionError
      ============2 failed, 1 passed, 4 warnings in 2.70 seconds ================
```

Пример

Три точки = три успешных теста

```
$ molecule verify
   =============== test session starts ==============================
    platform darwin -- Python 2.7.10, pytest-3.2.2, py-1.4.34, pluggy-0.4.0
    rootdir: /Users/test/mol/ansible/roles/db/molecule/default, inifile:
    plugins: testinfra-1.6.3
collected 3 items

    tests/test_default.py ...

============== 3 passed, 4 warnings in 3.02 seconds ===================
Verifier completed successfully.
```

Пример

```
$ molecule idempotence
--> Test matrix

└──── default
    └──── idempotence
--> Scenario: 'default'
--> Action: 'idempotence'
Idempotence completed successfully
```

Пример

```
$ molecule test

--> Test matrix

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

Матрица тестирования

- Можно менять порядок шагов тестирования
- Можно добавлять дополнительные шаги:
  - lint (yamllint, ansible-lint)
  - prepare
  - side-effect

Интеграция с CI

- Molecule не замена Jenkins, Bamboo, Gitlab CI, …
- Инструменты отлично дополняют друг друга
- Получаем общий набор тестов для локального и основного CI

Примеры pipelines

- Pipeline для Ansible роли
- Pipeline для инфраструктурного репозитория
- Pipeline для внешних зависимостей
- Pipeline для сборки образов
- Pipeline для создания окружений
- …

Практики из разработки и тестирования

- Инструменты:
  - IDE и Development kit
- Проблемы:
  - Разные инструменты
  - Разные версии инструментов
  - Разные подходы к использованию

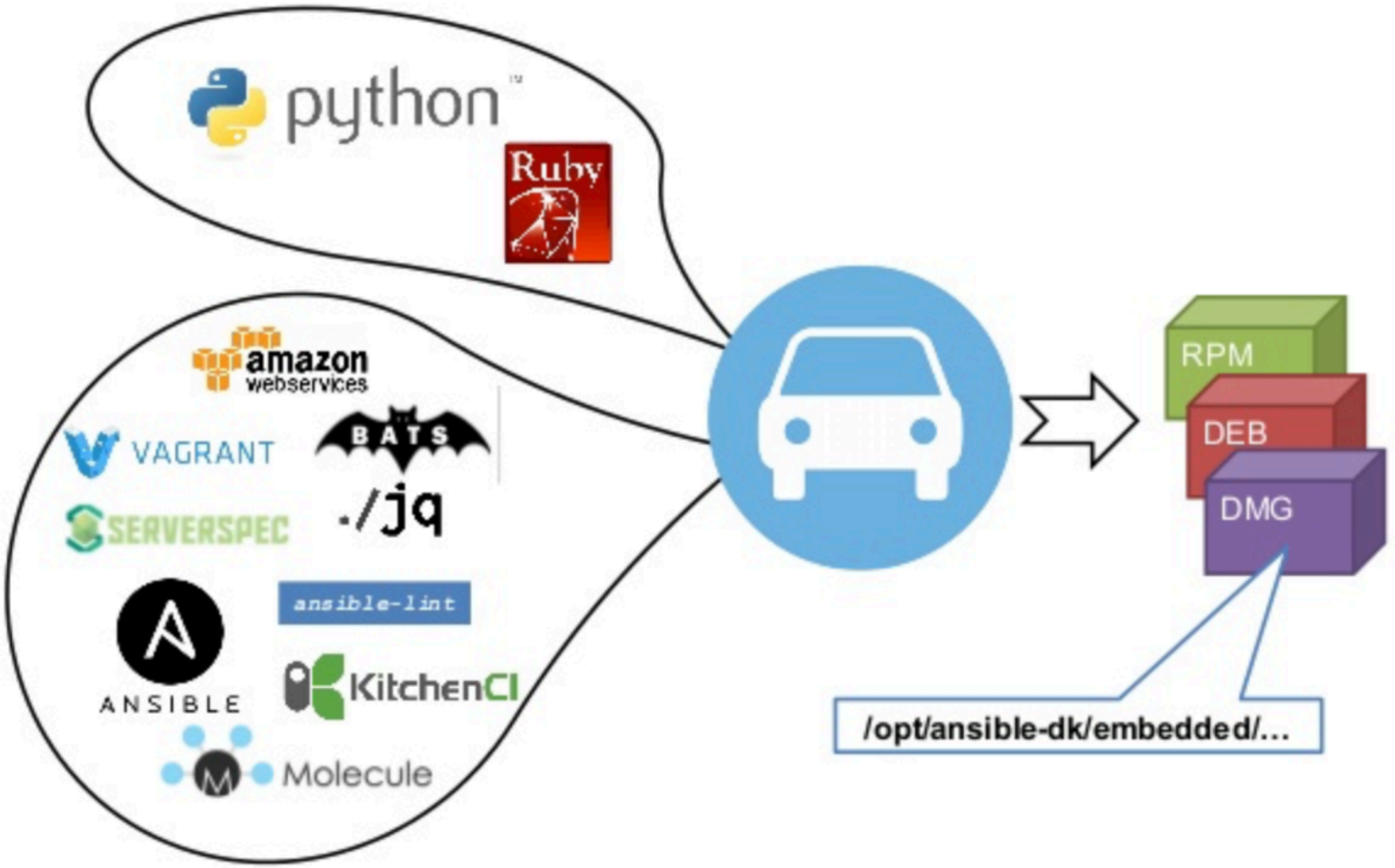
Development Kit

- Набор CLI инструментов, библиотек, языков, API и шаблонов
- Общее окружение для Linux, OS X, Windows
- Включает workflows и best practices
- Интеграция с IDE
- Часть CI

Варианты DK

- Chef DK, Chef workstation
- Puppet DK
- Ansible:
  - Официальный отсутствует
  - https://github.com/omniti-labs/ansible-dk
  - https://github.com/lean-delivery/ansible-development-kit
- Своя собственная реализация

ansible-dk


python
Ruby
amazon webservices
VAGRANT
BATS
./jq
SERVERSPEC
ansible-lint
ANSIBLE
KitchenCI
Molecule
RPM
DEB
DMG
/opt/ansible-dk/embedded/...

Практики из разработки

- Тестирование изменений
- Управление и синхронизация локальных окружений
- Внешние зависимости и управление артефактами
- Организация кода
- Версионирование и ветвление
- Ревью изменений
- Рефакторинг

Практики из разработки

- Ansible роль как артефакт:
  - Версионирование и неизменяемость
- Прогон артефакта по окружениям (dev-qa-preprod-prod)
- Тестовые данные
- Регулярное применение изменений и небольшие изменения

Практики из разработки

- Ревью кода:
  - Появляются общие подходы
  - Передача знаний внутри команды
- Делимся с сообществом:
  - Экспертиза и взгляд со стороны
  - Развитие и поддержка
  - Без тестов работать не будет

Итого

- Тестирование инфраструктурного кода
- Инструменты и примеры
- Практики из разработки

Ссылки

- Открытый урок Инфраструктура как код
- Книги Infrastructure as Code, Ansible for DevOps и Ansible Up and Running
- Официальная документация Ansible
- Статьи на Хабре:
  - https://habr.com/en/company/oleg-bunin/blog/431542/
  - https://habr.com/en/post/323472/
  - https://habr.com/en/post/358950/

Вопросы?