Testinfra test your infrastructure

Philippe Pepiot <phil@philpep.org>

Auteur et mainteneur de testinfra

Developpeur / Administrateur systèmes @

Logilab

Testinfra

- Outil pour tester l'état actuel et le comportement de vos serveurs
- En python
- https://testinfra.readthedocs.io
- https://github.com/philpep/testinfra
- Licence Apache 2

La petite histoire

```
- name: configure pgbouncer userlist
  lineinfile: dest=/etc/pgbouncer/userlist.txt
              regexp='^"user"' line='"user" "pass"'
```

Ansible 1.7.1:

```
"user" "pass"
```

Ansible 1.7.2:

```
user" "pass
```

Infrastructure as code

Infrastructure as code

- ansible, puppet, salt, chef, cfengine
- C'est du code !
- Comment le tester ?

Serveurs de tests, de preprod.

- Convient au sein d'une entreprise
- Tout le monde ne peut pas travailler dessus en même temps
- Ne convient pas aux projets avec des contributeurs externes (ex: ansible rôles, puppet modules, salt formulas)

Technologies

- Lancer le déploiement dans un environnement de test éphémère

Outils

- qemu https://www.qemu.org
- docker https://docker.com
- lxc https://linuxcontainers.org
- `$your_favorite_container_tool`
- Cloud Openstack / EC2 / GCE

Orchestration

- Lancer une VM / un container
- Deployer avec ansible, salt, puppet, chef, ...
- Vérifier

Outils

- Vagrant https://www.vagrantup.com
- Test-kitchen http://kitchen.ci
- Molecule https://molecule.readthedocs.io

Intégration continue

- Soumission de patch
- Processus de revue
- Tester les soumissions de patch automatiquement

Intégration continue

- Jenkins https://jenkins.io
- Travis https://travis-ci.org (on peut lancer des container docker dans travis: https://docs.travis-ci.com/user/docker/)

Vérifier le deploiement

- Si le deploiement ne plante pas c'est déjà bien
- Mais cela n'apporte pas de garantie sur le bon fonctionnement de l'infrastructure
- Le monitoring teste l'infrastructure en prod mais c'est déjà trop tard

Solution: **Écrire des tests à lancer après le déploiement**

Serverspec

- https://serverspec.org
- Première version en 2013
- Basé sur RSpec (Beharvior Driven Development en ruby)

```
describe package('httpd'),
    :if => os[:family] == 'redhat' do
  it { should be_installed }
end

describe package('apache2'),
    :if => os[:family] == 'ubuntu' do
  it { should be_installed }
end
```

Testinfra

- Première version en 2015
- En python, s'intègre avec pytest: https://pytest.org

```
def test_apache_installed(host):
    if host.system_info.distribution == 'redhat':
        pkgname = 'httpd'
    else:
        pkgname = 'apache2'
    assert host.package(pkgname).is_installed
```

pytest

- L'objet `host` est une fixture pytest définie dans testinfra qui représente par défaut l'hôte local
- Toutes les fonctions qui commencent par `test_` sont des tests pour pytest.
- Beaucoup de fonctionnalités: https://docs.pytest.org

Lancement des tests

```
$ py.test mytest.py
mytest.py::test_apache_installed[local://] PASSED
```

```
$ py.test mytest.py
[...]
mytest.py::test_apache_installed[local://] FAILED
[...]
  def test_apache_installed(host):
      if host.system_info.distribution == 'redhat':
          pkgname = 'httpd'
      else:
          pkgname = 'apache2'
>     assert host.package(pkgname).is_installed
E     AssertionError: assert False
E      +  where False = <package apache2>.is_installed
```

Testinfra

- Testinfra exécute des commandes localement ou sur un hôte distant
- Permet d'écrire des assertions sur l'état d'un serveur
- Abstraction du système avec des objets python

Connexions

- Accès à l'API via l'objet retourné par `testinfra.get_hosts()`
- Testinfra supporte différents types de connexion ("*backends*"): local, paramiko, docker, salt, ansible, kubectl, winrm
- Les connexions ont des options (utilisateur, sudo, ansible inventory)

https://testinfra.readthedocs.io/en/latest/backends.html

```
$ py.test --host=paramiko://user@host
test.py::test_postgres[pramiko://user@host] PASSED

$ py.test --host=docker://boring,docker://wozniak
test.py::test_postgres[docker://boring] PASSED
test.py::test_postgres[docker://wozniak] PASSED

$ py.test --host 'salt://web*'
$ py.test --host 'ansible://all?ansible_inventory=hosts'
```

Lancer des commandes

```
>>> import testinfra
>>> host = testinfra.get_host('ssh://somehost')
>>> host.run('echo foo;echo bar >/dev/stderr; exit 42')
CommandResult(rc=42, stdout='foo', stderr='bar')
```

Faire des assertions

```
>>> assert host.check_output('echo foo') == 'bar'
Traceback (most recent call last):
  [...]
AssertionError
```

Modules

package

- apt, rpm, *BSD

```
>>> host.package('nginx').is_installed
True
>>> host.package('nginx').version
'1.2.1-2.2+wheezy3'
```

service

- Systemd, Upstart, Sysv, *BSD

```
>>> host.service('nginx').is_enabled
True
>>> host.service('nginx').is_running
False
```

Modules

https://testinfra.readthedocs.io/en/latest/modules.html

ansible, file, group, interface, mount_point, package, pip_package, process, puppet_resource, facter, salt, service, socket, sudo, supervisor, sysctl, system_info, user

```
def test_postgres(host):
    postgresql = host.service('postgresql')
    assert postgresql.is_enabled
    assert postgresql.is_running

    with host.sudo('postgres'):
        assert host.check_output(
            "psql -tAc 'show shared_buffers'"
        ) == '256MB'

    assert sum([p.pmem for p in
                host.process.filter(user='postgres')
               ]) < 80
    assert len(host.socket('tcp://5432').clients) < 50
```

ansible

```
>>> host.ansible(
...     "apt", "name=nginx state=present")["changed"]
False
>>> host.ansible.get_variables()
{
    'inventory_hostname': 'localhost',
    'group_names': ['ungrouped'],
    'foo': 'bar',
}
```

Unittest

- Testinfra peut s'utiliser sans pytest
- On peut obtenir un objet `host` avec `testinfra.get_host()`
- Par exemple unittest:

```
import unittest
import testinfra

class TestTC(unittest.TestCase):

    def setUp(self):
        self.host = testinfra.get_host('local://')

    def test_postgres(self):
        self.assertTrue(
            self.host.service('postgres').is_enabled)

if __name__ == '__main__':
    unittest.main()
```

Test multi hôte

- On peut instancier plusieurs hôtes dans les tests
- Sans coût supplémentaire, il y a un cache des hôtes.

```
import testinfra

def test_same_users():
    srv0 = testinfra.get_host('paramiko://srv0')
    srv1 = testinfra.get_host('paramiko://srv1')
    assert srv0.check_output('getent passwd') == (
        srv1.check_output('getent passwd'))
```

Réutilisation des tests

- Certains tests peuvent être proches des tests de monitoring
- L'option `--nagios` permet à testinfra de se comporter comme une sonde nagios

```
$ testinfra -q --nagios
TESTINFRA OK - 1 passed, 0 failed, 0 skipped in 0.02 secc
```

Test de Dockerfile

Surcharge de la fixture `host`

```
import testinfra

@pytest.fixture(scope="session")
def host():
    check_output = testinfra.get_host(
        'local://').check_output
    check_output('docker build . -t myapp')
    docker_id = check_output('docker run -d myapp')
    yield testinfra.get_host(
        'docker://{}'.format(docker_id))
    check_output('docker rm -f %s', docker_id)

def test_myapp(host):
    # write assertions
```

Test-kitchen

- On peut utiliser testinfra en *verifier* de test-kitchen

```
verifier:
  name: shell
  command: testinfra --host="paramiko://
  ${KITCHEN_USERNAME}@${KITCHEN_HOSTNAME}:${KITCHEN_PORT}
  ?ssh_identity_file=${KITCHEN_SSH_KEY}"
  --junit-xml "junit-${KITCHEN_INSTANCE}.xml"
  "test/integration/${KITCHEN_SUITE}"
```

Vagrant

```
$ vagrant ssh-config > .vagrant/ssh-config
$ py.test --hosts=default \
    --ssh-config=.vagrant/ssh-config tests/
```

Molecule

- https://molecule.readthedocs.io
- https://github.com/metacloud/molecule
- Outil complet pour tester des rôles ansible
- Intègre testinfra nativement

Exemples de suite de tests

- https://github.com/ceph/ceph-ansible
- https://github.com/freedomofpress/securedrop

Autres projets

- serverspec (BDD, ruby)
- goss: https://github.com/aelsabbahy/goss en go, fichier de spec json, local only
- inspec, ruby orienté chef: https://www.inspec.io/

Roadmap

- Lancer des commandes en background

```
with host.run('journalctl -f') as output:
    testmyapp()
assert 'some log' in output.stdout
```

- Pouvoir maintenir des modules, connecteurs en dehors de testinfra
- Support MacOS / Windows ? (help wanted)

Questions ?

https://philpep.org/dl/testinfra_pyconfr_2017.pdf