# Xtesting

**README.md** (source from Functest-Xtesting)

## Xtesting in a nutshell

Xtesting is a simple framework to assemble sparse test cases and to accelerate the adoption of continuous integration best practices. By managing all the interactions with the components (test scheduler, test results database, artifact repository), it allows the developer to work only on the test suites without diving into CI/CD.

It asks for a few low constraints quickly achievable to verify multiple components in the same CI/CD toolchain. Even more, it brings the capability to run third-party test cases in our CI toolchains and then to also rate network functions by the coverage.

Please see the Katacoda scenarios to try Xtesting. You will love them!

## Write your own Xtesting driver

Note that running MongoDB 5.0+ requires *avx* CPU instruction set that is usually shipped in all recent *x86* hardware processors. Though, it may not be available in your virtualized environments. For example, Qemu *avx* support is only available since version 7.2 and must be explicitly enabled (e. g. with the argument *-cpu max*).

You can check the presence of the *avx* CPU instruction set on your processor with the following command.

```
grep '^processor\|^flags.* avx' /proc/cpuinfo
```

### dump all the following files in an empty dir

weather.py

```
#!/usr/bin/env python

# pylint: disable=missing-docstring

import json
import os
import sys
import time

import requests

from xtesting.core import testcase


class Weather(testcase.TestCase):

    url = "https://samples.openweathermap.org/data/2.5/weather"
    city_name = "London,uk"
    app_key = "439d4b804bc8187953eb36d2a8c26a02"

    def run(self, **kwargs):
        try:
            self.start_time = time.time()
            req = requests.get("{}?q={}&&appid={}".format(
                self.url, self.city_name, self.app_key))
            req.raise_for_status()
            data = req.json()
            os.makedirs(self.res_dir, exist_ok=True)
            with open('{}/dump.txt'.format(self.res_dir), 'w+') as report:
                json.dump(data, report, indent=4, sort_keys=True)
            for key in kwargs:
                if data["main"][key] > kwargs[key]:
                    self.result = self.result + 100/len(kwargs)
            self.stop_time = time.time()
        except Exception:  # pylint: disable=broad-except
            print("Unexpected error:", sys.exc_info()[0])
            self.result = 0
            self.stop_time = time.time()
```

setup.py

```
#!/usr/bin/env python

# pylint: disable=missing-docstring

import setuptools

setuptools.setup(
    setup_requires=['pbr>=2.0.0'],
    pbr=True)
```

setup.cfg

```
[metadata]
name = weather
version = 1

[files]
packages = .

[entry_points]
xtesting.testcase =
    weather = weather:Weather
```

requirements.txt

```
xtesting
requests!=2.20.0,!=2.24.0 # Apache-2.0
```

testcases.yaml

```
---
tiers:
    -
        name: simple
        order: 0
        description: ''
        testcases:
            -
                case_name: humidity
                project_name: weather
                criteria: 100
                blocking: true
                clean_flag: false
                description: ''
                run:
                    name: weather
                    args:
                        humidity: 80
            -
                case_name: pressure
                project_name: weather
                criteria: 100
                blocking: true
                clean_flag: false
                description: ''
                run:
                    name: weather
                    args:
                        pressure: 1000
            -
                case_name: temp
                project_name: weather
                criteria: 100
                blocking: true
                clean_flag: false
                description: ''
                run:
                    name: weather
                    args:
                        temp: 280
    -
        name: combined
        order: 1
        description: ''
        testcases:
            -
                case_name: half
                project_name: weather
                criteria: 50
                blocking: true
                clean_flag: false
                description: ''
                run:
                    name: weather
                    args:
                        humidity: 90
                        pressure: 1000
                        temp: 280
```

Dockerfile

```
FROM alpine:3.18

ADD . /src/
RUN apk --no-cache add --update python3 py3-pip py3-wheel git py3-lxml && \
    git init /src && pip3 install /src
COPY testcases.yaml /etc/xtesting/testcases.yaml
CMD ["run_tests", "-t", "all"]
```

site.yml

```
---
- hosts:
    - 127.0.0.1
  roles:
    - role: collivier.xtesting
      project: weather
      registry_deploy: true
      repo: 127.0.0.1
      dport: 5000
      suites:
        - container: weather
          tests:
            - humidity
            - pressure
            - temp
            - half
```

## make world

Deploy your own Xtesting toolchain

```
virtualenv xtesting -p python3 --system-site-packages
. xtesting/bin/activate
pip install ansible
ansible-galaxy install collivier.xtesting
ansible-galaxy collection install ansible.posix community.general community.grafana \
    community.kubernetes community.docker community.postgresql
ansible-playbook site.yml
deactivate
rm -r xtesting
```

Build your container

```
sudo docker build -t 127.0.0.1:5000/weather .
```

Publish your container on your local registry

```
sudo docker push 127.0.0.1:5000/weather
```

## play

Jenkins is accessible via http://127.0.0.1:8080 and you can identify yourself as admin to be allowed to trigger a build:

- login: admin
- password: admin

The default Jenkins view lists all the Jenkins jobs. You can easily find your main job, weather-latest-daily, via the Jenkins view named weather.

You're ready to start a new build of weather-latest-daily without changing the default parameters.

The test case is executed after a few seconds and all the test outputs are accessible via the console icons. If you open the weather-127_0_0_1-weather-latest-humidity-run, you will first read:

- the test output highlighting its status
- a link to the test database where its results are stored
- a couple of links to its artifacts automatically published

A zip file dumping all test campaign data is printed in the weather-latest-zip console.

## That's all folks!