testcontainers Documentation

Release 2.0.0

Sergey Pirogov

May 05, 2020

Contents

Python port for testcontainers-java that allows using docker containers for functional and integration testing. Testcontainers-python provides capabilities to spin up docker containers (such as a database, Selenium web browser, or any other container) for testing.

Currently available features:

- Selenium Grid containers
- Selenium Standalone containers
- MySql Db container
- MariaDb container
- OracleDb container
- PostgreSQL Db container
- Microsoft SQL Server container
- Generic docker containers

CHAPTER 1

Installation

The testcontainers package is available from PyPI, and it can be installed using `pip`. Depending on which containers are needed, you can specify additional dependencies as extras:

```
# Install without extras
pip install testcontainers
# Install with one or more extras
pip install testcontainers[mysql]
pip install testcontainers[mysql,oracle]
```

CHAPTER 2

Basic usage

```
import sqlalchemy
from testcontainers.mysql import MySqlContainer

with MySqlContainer('mysql:5.7.17') as mysql:
    engine = sqlalchemy.create_engine(mysql.get_connection_url())
    version, = engine.execute("select version()").fetchone()
    print(version)  # 5.7.17
```

The snippet above will spin up a MySql database in a container. The `get_connection_url()` convenience method returns a `sqlalchemy` compatible url we use to connect to the database and retrieve the database version.

More extensive documentation can be found at Read The Docs.

CHAPTER 3

Setting up a development environment

We recommend you use a virtual environment for development. Note that a python version `>=3.5` is required. After setting up your virtual environment, you can install all dependencies and test the installation by running the following snippet.

```
pip install -r requirements/$(python -c 'import sys; print("%d.%d" % sys.version_
↪info[:2])').txt
pytest -s
```

3.1 Adding requirements

We use `pip-tools` to resolve and manage dependencies. If you need to add a dependency to testcontainers or one of the extras, run `pip install pip-tools` followed by `make requirements` to update the requirements files.

CHAPTER 4

Usage modes

4.1 Database containers

Allows to spin up database images such as MySQL, PostgreSQL, MariaDB, Oracle XE, or MongoDb.

class testcontainers.mysql.**MySqlContainer**(*image='mysql:latest'*, ***kwargs*)
MySql database container.

Example

The example will spin up a MySql database to which you can connect with the credentials passed in the constructor. Alternatively, you may use the `get_connection_url()` method which returns a sqlalchemy-compatible url in format `dialect+driver://username:password@host:port/database`.

```
with MySqlContainer('mysql:5.7.17') as mysql:
    e = sqlalchemy.create_engine(mysql.get_connection_url())
    result = e.execute("select version()")
    version, = result.fetchone()
```

class testcontainers.mysql.**MariaDbContainer**(*image='mariadb:latest'*, ***kwargs*)
Maria database container, a commercially-supported fork of MySql.

Example

```
with MariaDbContainer("mariadb:latest") as mariadb:
    e = sqlalchemy.create_engine(mariadb.get_connection_url())
    result = e.execute("select version()")
```

class testcontainers.postgres.**PostgresContainer**(*image='postgres:latest'*)
Postgres database container.

Example

The example spins up a Postgres database and connects to it using the `psycopg` driver.

```
with PostgresContainer("postgres:9.5") as postgres:
    e = sqlalchemy.create_engine(postgres.get_connection_url())
    result = e.execute("select version()")
```

class testcontainers.oracle.**OracleDbContainer**(*image='wnameless/oracle-xe-11g-r2:latest'*)

Oracle database container.

Example

```
with OracleDbContainer():
    e = sqlalchemy.create_engine(oracle.get_connection_url())
    result = e.execute("select 1 from dual")
```

class testcontainers.elasticsearch.**ElasticSearchContainer**(*image='elasticsearch:7.5.0'*, *port_to_expose=9200*)

ElasticSearch container.

Example

```
with ElasticSearchContainer() as es:
    connection_url = es.get_url()
```

class testcontainers.mongodb.**MongoDbContainer**(*image='mongodb:latest'*)

Mongo document-based database container.

Example

```
with MongoDbContainer("mongo:latest") as mongo:
    db = mongo.get_connection_client().test
    # Insert a database entry
    result = db.restaurants.insert_one(
        {
            "address": {
                "street": "2 Avenue",
                "zipcode": "10075",
                "building": "1480",
                "coord": [-73.9557413, 40.7720266]
            },
            "borough": "Manhattan",
            "cuisine": "Italian",
            "name": "Vella",
            "restaurant_id": "41704620"
        }
    )
    # Find the restaurant document
    cursor = db.restaurants.find({"borough": "Manhattan"})
    for document in cursor:
        # Do something interesting with the document
```

class testcontainers.mssql.**SqlServerContainer**(*image='mcr.microsoft.com/mssql/server:2019-latest'*, *user='SA'*, *password=None*, *port=1433*, *dbname='tempdb'*, *driver='ODBC Driver 17 for SQL Server'*)

Microsoft Sql Server database container.

Example

```
with SqlServerContainer() as mssql:
    e = sqlalchemy.create_engine(mssql.get_connection_url())
    result = e.execute("select @@VERSION")
```

Notes

Requires ODBC Driver 17 for SQL Server.

4.2 Selenium containers

Allows to spin up selenium containers for testing with browsers.

class testcontainers.selenium.**BrowserWebDriverContainer**(*capabilities*, *image=None*)

Selenium browser container for Chrome or Firefox.

Example

```
from selenium.webdriver import DesiredCapabilities

with BrowserWebDriverContainer(DesiredCapabilities.CHROME) as chrome:
    webdriver = chrome.get_driver()
    webdriver.get("http://google.com")
    webdriver.find_element_by_name("q").send_keys("Hello")
```

You can easily change browser by passing `DesiredCapabilities.FIREFOX` instead.

4.3 Docker compose support

Allows to spin up services configured via `docker-compose.yml`.

class testcontainers.compose.**DockerCompose**(*filepath*, *compose_file_name='docker-compose.yml'*, *pull=False*)

Docker compose containers.

Example

```
with DockerCompose("/home/project", pull=True) as compose:
    host = compose.get_service_host("hub", 4444)
    port = compose.get_service_port("hub", 4444)
    driver = webdriver.Remote(
        command_executor=("http://{}:{}/wd/hub".format(host,port)),
        desired_capabilities=CHROME,
    )
    driver.get("http://automation-remarks.com")
```

```
hub:
image: selenium/hub
ports:
- "4444:4444"
firefox:
image: selenium/node-firefox
links:
    - hub
expose:
    - "5555"
chrome:
image: selenium/node-chrome
links:
    - hub
expose:
    - "5555"
```

4.4 Google Cloud Emulators

Allows to spin up google cloud emulators, such as PubSub.

class testcontainers.google.**PubSubContainer**(*image='google/cloud-sdk:latest'*, *project='test-project'*, *port=8432*)

PubSub container for testing managed message queues.

Example

The example will spin up a Google Cloud PubSub emulator that you can use for integration tests. The `pubsub` instance provides convenience methods `get_publisher` and `get_subscriber` to connect to the emulator without having to set the environment variable `PUBSUB_EMULATOR_HOST`.

```
def test_docker_run_pubsub():
    config = PubSubContainer('google/cloud-sdk:latest')
    with config as pubsub:
        publisher = pubsub.get_publisher()
        topic_path = publisher.topic_path(pubsub.project, "my-topic")
        topic = publisher.create_topic(topic_path)
```

Python Module Index

t

Index