testcontainers Documentation

Release 2.0.0

Sergey Pirogov

Jun 14, 2022

CONTENTS

1 Installation **3**

2 Basic usage **5**

3 Usage within Docker (i.e. in a CI) **7**

4 Setting up a development environment **9**
- 4.1 Adding requirements . . . 9
- 4.2 Contributing a new container . . . 9

5 Usage modes **11**
- 5.1 Database containers . . . 11
- 5.2 Selenium containers . . . 14
- 5.3 Docker Compose Support . . . 14
- 5.4 Google Cloud Emulators . . . 16

Python Module Index **17**

Index **19**

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
- Neo4j container
- OracleDb container
- PostgreSQL Db container
- ClickHouse container
- Microsoft SQL Server container
- Generic docker containers
- LocalStack
- RabbitMQ
- Keycloak

CHAPTER

ONE

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

TWO

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

THREE

USAGE WITHIN DOCKER (I.E. IN A CI)

When trying to launch a testcontainer from within a Docker container two things have to be provided:

1. The container has to provide a docker client installation. Either use an image that has docker pre-installed (e.g. the official docker images) or install the client from within the *Dockerfile* specification.
2. The container has to have access to the docker daemon which can be achieved by mounting */var/run/docker.sock* or setting the *DOCKER_HOST* environment variable as part of your *docker run* command.

CHAPTER

FOUR

SETTING UP A DEVELOPMENT ENVIRONMENT

We recommend you use a virtual environment for development. Note that a python version >=3.6 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
```

4.1 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, modify the `setup.py` as well as the `requirements.in` accordingly and then run `pip install pip-tools` followed by `make requirements` to update the requirements files.

4.2 Contributing a new container

You can contribute a new container in three steps:

1. Create a new module at `testcontainers/[my fancy container].py` that implements the new functionality.
2. Create a new test module at `tests/test_[my fancy container].py` that tests the new functionality.
3. Add `[my fancy container]` to the list of test components in the GitHub Action configuration at `.github/workflows/main.yml`.

CHAPTER

FIVE

USAGE MODES

5.1 Database containers

Allows to spin up database images such as MySQL, PostgreSQL, MariaDB, Oracle XE, MongoDb, ClickHouse or Neo4j.

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'*, *port=5432*, *user=None*, *password=None*, *dbname=None*, *driver='psycopg2'*, ***kwargs*)

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'*, ***kwargs*)

Oracle database container.

Example

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

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

ElasticSearch container.

Example

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

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

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"})
```

(continues on next page)

(continued from previous page)

```
    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'*, *dialect='mssql+pymssql'*, ***kwargs*)

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.

class testcontainers.clickhouse.**ClickHouseContainer**(*image='clickhouse/clickhouse-server:latest'*, *port=9000*, *user=None*, *password=None*, *dbname=None*)

ClickHouse database container.

Example

The example spins up a ClickHouse database and connects to it using the `clickhouse-driver`.

```
with ClickHouseContainer("clickhouse/clickhouse-server:21.8") as clickhouse:
    with clickhouse_driver.Client.from_url(self.get_connection_url()) as client:
        result = client.execute("SELECT version()")
```

class testcontainers.neo4j.**Neo4jContainer**(*image='neo4j:latest'*, ***kwargs*)

Neo4j Graph Database (Standalone) database container.

Example

::

with Neo4jContainer() as neo4j:

with neo4j.get_driver() as driver:

with driver.session() as session: result = session.run("MATCH (n) RETURN n LIMIT 1") record = result.single()

5.2 Selenium containers

Allows to spin up selenium containers for testing with browsers.

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

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.

5.3 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*, *build=False*, *env_file=None*)

Manage docker compose environments.

Parameters

- **filepath** (*str*) – The relative directory containing the docker compose configuration file
- **compose_file_name** (*str*) – The file name of the docker compose configuration file
- **pull** (*bool*) – Attempts to pull images before launching environment
- **build** (*bool*) – Whether to build images referenced in the configuration file
- **env_file** (*str*) – Path to an env file containing environment variables to pass to docker compose

Example

```
with DockerCompose("/home/project",
                   compose_file_name=["docker-compose-1.yml", "docker-compose-2.yml
↪"],
                   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")
```

(continues on next page)

(continued from previous page)

```
    stdout, stderr = compose.get_logs()
    if stderr:
        print("Errors\n:{}".format(stderr))
```

```
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"
```

docker_compose_command()

Returns command parts used for the docker compose commands

Returns The docker compose command parts

Return type list[str]

exec_in_container(*service_name*, *command*)

Executes a command in the container of one of the services.

Parameters

- **service_name** (*str*) – Name of the docker compose service to run the command in
- **command** (*list[str]*) – The command to execute

Returns stdout, stderr, return code

Return type tuple[str, str, int]

get_logs()

Returns all log output from stdout and stderr

Returns stdout, stderr

Return type tuple[bytes, bytes]

get_service_host(*service_name*, *port*)

Returns the host for one of the services.

Parameters

- **service_name** (*str*) – Name of the docker compose service
- **port** (*int*) – The internal port to get the host for

Returns The hostname for the service

Return type str

get_service_port(*service_name*, *port*)

Returns the mapped port for one of the services.

Parameters

- **service_name** (*str*) – Name of the docker compose service
- **port** (*int*) – The internal port to get the mapping for

Returns The mapped port on the host

Return type str

start()

Starts the docker compose environment.

stop()

Stops the docker compose environment.

wait_for(*url*)

Waits for a response from a given URL. This is typically used to block until a service in the environment has started and is responding. Note that it does not assert any sort of return code, only check that the connection was successful.

Parameters **url** (*str*) – URL from one of the services in the environment to use to wait on

5.4 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*, ***kwargs*)

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

testcontainers.compose, 14
testcontainers.google, 16
testcontainers.selenium, 13

INDEX

B

BrowserWebDriverContainer (*class in testcontainers.selenium*), 14

C

ClickHouseContainer (*class in testcontainers.clickhouse*), 13

D

docker_compose_command() (*testcontainers.compose.DockerCompose method*), 15
DockerCompose (*class in testcontainers.compose*), 14

E

ElasticSearchContainer (*class in testcontainers.elasticsearch*), 12
exec_in_container() (*testcontainers.compose.DockerCompose method*), 15

G

get_logs() (*testcontainers.compose.DockerCompose method*), 15
get_service_host() (*testcontainers.compose.DockerCompose method*), 15
get_service_port() (*testcontainers.compose.DockerCompose method*), 15

M

MariaDbContainer (*class in testcontainers.mysql*), 11
module
    testcontainers.compose, 14
    testcontainers.google, 16
    testcontainers.selenium, 13
MongoDbContainer (*class in testcontainers.mongodb*), 12
MySqlContainer (*class in testcontainers.mysql*), 11

N

Neo4jContainer (*class in testcontainers.neo4j*), 13

O

OracleDbContainer (*class in testcontainers.oracle*), 12

P

PostgresContainer (*class in testcontainers.postgres*), 11
PubSubContainer (*class in testcontainers.google*), 16

S

SqlServerContainer (*class in testcontainers.mssql*), 13
start() (*testcontainers.compose.DockerCompose method*), 16
stop() (*testcontainers.compose.DockerCompose method*), 16

T

testcontainers.compose
    module, 14
testcontainers.google
    module, 16
testcontainers.selenium
    module, 13

W

wait_for() (*testcontainers.compose.DockerCompose method*), 16