**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

Курсовая РАБОТА

**по дисциплине «Технологии автоматизации процесса разработки программного обеспечения»**

**Тема: Использование Docker**

**Вариант 21**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8310 |  | Иванов Я.А. |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

2023

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Иванов Ярослав | | |
| Группа 8310 | | |
| Тема работы: Использование Docker | | |
| Исходные данные:  Требуется реализовать конфигурацию docker-compose, состоящую из двух контейнеров – с приложением и с тестами. | | |
| Содержание пояснительной записки:  Содержание, Введение, Постановка задачи, Описание Dockerfile, Описание скриптов запуска тестов, Описание конфигурации docker-compose, Заключение, Список использованных источников. Приложение. | | |
| Предполагаемый объем пояснительной записки:  Не менее 15 страниц. | | |
| Дата выдачи задания: | | |
| Дата сдачи реферата: | | |
| Дата защиты реферата: | | |
| Студент |  | Иванов Я.А. |
| Преподаватель |  | Заславский М.М. |

**Аннотация**

В результате данной работы была реализована конфигурация docker-compose, состоящая из двух контейнеров, и соответствующие им dockerfile. Один контейнер запускает предоставленное приложение, другой – тесты. Тестирование включает анализатор кода pylint, запуск реализованных интеграционных тестов, бьютификатор HTML файлов и selenium тесты.

**Summary**

This work implemented a Docker-compose configuration consisting of two containers and their corresponding Dockerfiles. One container runs a provided application, while the other runs tests that include a PyLint code analyzer, integration tests, selenium test and an HTML beautifier.

Содержание

[Введение 5](#_Toc134223602)

[Постановка задачи 6](#_Toc134223603)

[1. Описание Dockerfile 8](#_Toc134223604)

[1.1. Dockerfile для app-контейнера 8](#_Toc134223605)

[1.2. Dockerfile для tester-контейнера 8](#_Toc134223606)

[2. Описание скриптов запуска тестов 10](#_Toc134223607)

[2.1. Бьютификация HTML 10](#_Toc134223608)

[2.2. Статический анализ по 10 критериям 10](#_Toc134223609)

[2.3. Интеграционный тест 10](#_Toc134223610)

[2.4. Selenium тест 10](#_Toc134223611)

[2.5. Запуск тестов 10](#_Toc134223612)

[3. Описание Docker-compose конфигурации 12](#_Toc134223613)

[Заключение 13](#_Toc134223614)

[Список использованных источников 14](#_Toc134223615)

# Введение

В данной работе реализована конфигурация docker-compose, состоящая из двух контейнеров, и dockerfile для их создания. Один контейнер используется для запуска веб-сервера, другой – для запуска тестов. Для запуска тестов используется pytest.

# Постановка задачи

Необходимо реализовать конфигурацию docker-compose, состоящую из двух узлов: с приложением и с тестами для него. При этом необходимо выполнить следующие задачи:

1. Добавить в тестирование бьютификацию HTML файлов;
2. Проанализировать по десяти существующим критериям статического анализа;
3. Написать интеграционный тест, проверяющий загрузку файла.
4. Написать selenium тест, проводящий правку HTML страницы и проверку добавленного элемента.
5. Записывать каждый этап тестирования в docker log, а также оба потока в один файл
6. Добавить возможность передачи ssh ключа;
7. Ограничить максимальное потребление процессов.

Требования к docker-compose:

* Минимальная версия docker compose version 1.27.4, build 40524192
* Все должно собираться по команде docker-compose build без sudo
* Не использовать тип сети HOST
* Не отрывать лишних (непредусмотренных заданием) портов
* Не использовать порты хост-машины ⇐1024

Требования к Dockerfile:

* Минимальная версия докера Docker version 19.03.13, build 4484c46d9d
* Базовый образ ubuntu:22.04
* Не использовать Expose
* При установке любых пакетов и программ (в том числе в requirements) ВСЕГДА указывать версии
* Ограничить установку зависимостей apt одной строкой (один RUN)
* Если настройка одной части приложения состоит из нескольких команд → необходимо разместить их в одном слое (в одном RUN)

# Описание Dockerfile

## Dockerfile для app-контейнера

В качестве базового используется образ ubuntu:22.04.

Устанавливаются зависимости:

* git – для скачивания кода приложения;
* python – для его запуска;
* pip – для установки зависимостей.

Загружается код приложения, рабочей директорией устанавливается директория с кодом приложения.

Устанавливаются переменные окружения Flask – хост и файл сервера

(main.py).

В контейнер копируется файл со списком python-зависимостей, после чего с помощью pip выполняется установка указанных библиотек.

Указывается точка входа – код приложения.

## Dockerfile для tester-контейнера

В качестве базового используется образ ubuntu:22.04.

Устанавливаются зависимости:

* git – для скачивания кода приложения;
* python – для запуска тестов;
* pip – для установки зависимостей;
* wget – для скачивания firefox и geckodriver;
* xvfb – для запуска firefox без графического интерфейса;
* libgtk, libdbus и libasound – зависимости, необходимые для запуска firefox;

Выполняется установка firefox и geckodriver, необходимых для запуска selenium-тестов.

Загружается код приложения, рабочей директорией устанавливается директория с кодом приложения.

В контейнер копируются написанные тесты и файл с перечислением python-зависимостей для этих тестов. Выполняется установка библиотек.

Устанавливается точка входа – http-сервер.

# Описание скриптов запуска тестов

## Бьютификация HTML

Для бьютификации HTML кода используется библиотека css-html-prettify, которая запускается python файлом. Библиотека редактирует файлы HTML и CSS, приводя к единому стилю форматирования. Передается папка templates, в которой содержатся HTML файлы app. Бьютификация проводится в контейнере tester – не влияет на код, содержащийся и запускаемый в app контейнере.

## Статический анализ по 10 критериям

В настройке pylint установлена проверка по критериям: main, basic, format, classes, design, imports, typecheck, variables, refactoring, string.

## Интеграционный тест

В интеграционном тесте проверяется загрузка файла.

## Selenium тест

Тестирование проводится с помощью драйвера браузера Firefox. Производить открытие веб-страницы приложения, запущенного в контейнере app, по короткой ссылке (через сеть docker). Далее с помощью JS-инъекции execute\_script выполнять модификацию HTML страницы – добавляется кнопка с уникальным id. Далее происходит проверка добавленного элемента путем сверки текста, который в нем должен содержаться.

## Запуск тестов

Запуск тестов производится с помощью .sh скриптов.

* pylint.sh– запускает статическую проверку по 10 критериям
* integration.sh– запускает интеграционный тест
* prettify.sh– запускает бьютификатор HTML

Вывод логов осуществляется в один файл из двух потоков:

2>&1 | tee -a /proc/1/fd/1 /logs/tester-logs.txt

Также к логам добавляется дата и время.

Используются флаги Pytest для выключения предупреждений, а также вывода не только ошибок, но и списка тестов, которые прошли успешно, а также короткий формат сообщений «-rpP --tb=short».

# Описание Docker-compose конфигурации

Конфигурация docker-compose задается в файле docker-compose.yml. В нем описывается два узла: app и testser.

Узел app:

* Используется образ Docker\_app;
* Подключается файл с переменными среды;
* Соотносится внешний порт 127.0.0.1:5000 с портом 5000 в контейнере;
* Соотносится порт 2222:22 дня внешнего доступа по ssh
* Ограничивается максимальное потребление процессов в 50.

Узел tester:

* Устанавливается зависимость от сервиса app;
* Используется образ Docker\_tester;
* Устанавливается volume для передачи логов на хост машину - ./logs:/logs

# Заключение

В ходе выполнения данной работы были изучены возможности docker и docker-compose. Для этого была реализована конфигурация, состоящая из двух контейнеров: один – для запуска приложения, другой – для запуска тестов.

В ходе работы были реализованы различные виды тестов, решены задачи отображения вывода процессов в контейнере в docker log и ограничения максимального потребления процессов.

# Список использованных источников

1. Документация по Dockerfile : сайт. – URL: https://docs.docker.com/engine/reference/builder/ (дата обращения: 10.09.2023)
2. Документация по Docker-compose : сайт. – URL: https://docs.docker.com/compose/compose-file/ (дата обращения: 10.09.2023)
3. Документация pytest : сайт. – URL: https://docs.pytest.org/en/7.1.x/contents.html (дата обращения: 10.09.2023)
4. Документация pylint : сайт. – URL: https://docs.pylint.org/ (дата обращения: 10.09.2023)