**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: Разработка системы автоматизированного тестирования веб-приложений**

**Вариант 13**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9303 |  | Шахин Н.С. |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

2024

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Шахин Н.С. | | |
| Группа 9303 | | |
| Тема работы: Разработка системы автоматизированного тестирования веб-приложений | | |
| Исходные данные:  Необходимо реализовать docker-compose конфигурацию из двух узлов:   * app – контейнер с существующим демонстрационным веб-приложением * tester – контейнер для запуска всех тестов | | |
| Содержание пояснительной записки:  Содержание; Введение; Постановка задачи; Описание Dockerfile; Описание скриптов запуска тестов; Описание docker-compose конфигурации; Заключение; Список использованных источников. | | |
| Предполагаемый объем пояснительной записки:  Не менее 12 страниц. | | |
| Дата выдачи задания: | | |
| Дата сдачи реферата: | | |
| Дата защиты реферата: | | |
| Студент |  | Шахин Н.С. |
| Преподаватель |  | Заславский М.М. |

**Аннотация**

В данной курсовой работе описана конфигурация системы для автоматизированного тестирования веб-приложений – демонстрационного и тестового экземпляра ИС ИОТ. Система состоит из двух контейнеров: в одном запускается демонстрационное веб-приложение, второй используется для запуска тестов.

**Summary**

This course work describes the configuration of the system for automated testing of web applications – a demo and test instance of the IS IOT. The system consists of two containers: one runs a demo web application, the second is used to run tests.**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 5 |
|  | Постановка задачи | 6 |
| 1. | Описание Dockerfile | 8 |
| 1.1. | Dockerfile для app-контейнера | 8 |
| 1.2. | Dockerfile для tester-контейнера | 8 |
| 2. | Описание тестов | 10 |
| 2.1. | Этап форматирования | 10 |
| 2.2. | Этап интеграционного тестирования | 10 |
| 3. | Описание docker-compose | 11 |
|  | Заключение | 12 |
|  | Список использованных источников | 13 |
|  | Приложение А. Исходный код проекта | 14 |

**введение**

Целью данной работы является реализация docker-compose конфигурации, предназначенной для сборки и запуска контейнеров app и tester. Контейнеры по отдельности выполняют задачи, включающие в себя запуск демонстрационного веб-приложения, а также тестирование данного веб-приложения.

**Постановка задачи**

Необходимо реализовать docker-compose конфигурацию из двух узлов:

* app - контейнер с существующим демонстрационным веб-приложением.
  + Устанавливать приложение необходимо скачивая репозиторий и копируя файлы из него при сборке вашего контейнера:)
  + Чтобы все заработало, вам придется потратить время и поразбираться - из коробки может не работать.
  + Возможно, вам для выполнения заданий потребуются фиксы в исходник - делайте для них патчи
  + Корнем дерева процессов выступает запущенное веб-приложение
* tester - контейнер для запуска **всех** тестов (состав и особенности тестов задаются в таблице вариантов)
  + Корнем дерева процессов выступает стандартный python http сервер (python -m http.server 3000)
  + Этот сервер должен быть запущен в каталоге контейнера, где будет происходить работа тестовых скриптов
  + Тестовые скрипты запускаются через docker exec

При этом при разработке необходимо учесть следующие требования:

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

Параметры конфигурации, заданные для 13 варианта:

|  |  |
| --- | --- |
| Параметр | Требование |
| Проверка на соответствие стилю кодирования / бьютификация | Проверка на pep8 |
| Статический анализ | Создание своего критерия и проверка только по нему |
| Интеграционные тесты | Проверка на коды возврата |
| Selenium-тесты | Проверка формы "Назначение прав и ролей" |
| Внешний SSH доступ в контейнеры | В tester - по паролю |
| Вывод логов работы tester | Каждый этап тестирования - в docker log (stdout + stderr) + в отдельный файл оба потока по каждому виду тестирования |
| Передача параметров в конфигурацию через .env | Порт для веб-сервера |
| Органичения ресурсов | настройки | ОЗУ |

**1. Описание Dockerfile**

**1.1. Dockerfile для app-контейнера**

Последовательность инструкций создания образа app-контейнера:

1. Базовый образ - ubuntu:22.04.
2. Обновляются пакетные списки и устанавливаются необходимые apt-зависимости:
   * git
   * python3
   * python3-pip
3. Клонируется репозиторий с демонстрационным веб-приложением и устанавливается рабочая директория внутри этого репозитория
4. Копируется реализованный patch-файл, изменяющий main.py для корректной работы веб-приложения.
5. Устанавливаются необходимые зависимости Python, используемые в веб-приложении:
6. Задается точка входа для контейнера, запускающая веб-приложение.

**1.2. Dockerfile для tester-контейнера**

Последовательность инструкций создания образа tester-контейнера:

1. Базовый образ - ubuntu:22.04.
2. Обновляются пакетные списки и устанавливаются необходимые apt-зависимости:

* git
* openssh-server
* pwgen
* python3
* python3-pip

1. Клонируется демонстрационный проект из репозитория
2. Устанавливаются необходимые зависимости Python, используемые в тестах.
3. Копируются файлы из директории /tests внутрь контейнера
4. Производится настройка конфигурации SSH сервера для разрешения входа под пользователем root. Генерируется пароль.
5. Задается точка входа для контейнера, запускающая веб-сервер http на порту 3000 и ssh сервер.

**2. Описание скриптов запуска тестов**

**2.1. Скрипт run\_tests.sh для запуска этапов тестирования**

Запуск этапов тестирования возможен с использованием реализованного скрипта run\_tests.sh, запускающего все этапы совместно.

В каждой из функций в первую очередь выводится информация об этапе, после чего выполняются необходимые для запуска команды. Результаты выполнения с помощью команды tee перенаправляются в файл tester-logs.log, а также в stdout контейнера.

**2.2. Этап форматирования**

Для выполнения форматирования используется утилита pep8. Форматирование производится в соответствии с PEP 8.

**2.3. Этап интеграционного тестирования**

Для запуска интеграционных тестов используется фреймворк pytest. Запускается скрипт integration.py.

**3. Описание docker-compose конфигурации**

Конфигурация docker-compose описывается в файле docker-compose.yml и включает в себя описание запуска двух контейнеров – app (Hukumka-app) с помощью образа из Dockerfile\_app и tester (Hukumka-tester) с помощью образа из Dockerfile\_tester.

В процессе запуска контейнера с веб-приложением устанавливается ограничение на максимальный объём ОЗУ – 230 Мб, передаётся .env файл с переменными окружения, а также пробрасываются порты из контейнера на хост-машину. "127.0.0.1:${APP\_PORT}:5000" – порт веб-приложения внутри контейнера (5000) становится доступен на хост-машине по порту, указанному в переменных окружения (APP\_PORT).

В процессе запуска контейнера с тестирующими скриптами внутрь контейнера передается .env файл с переменными окружения, а также пробрасываются порты. "127.0.0.1:2222:22" – порт SSH-сервера внутри контейнера (22) преобразуется в 2222 порт на хост-машине для возможности дальнейшего получения доступа по SSH.

Для выполнения тестирования необходимо наличие запущенного контейнера app, соответствующая инструкция указана для контейнера tester.

**заключение**

По итогу выполнения курсовой работы были изучены технологии docker и docker-compose, применены на практике при реализации конфигурации из двух контейнеров – app для запуска приложения и SSH-сервера и tester для запуска процесса тестирования.

Процесс тестирования состоит из нескольких этапов, включая форматирование с использованием pep8, и интеграционное тестирование демонстрационного веб-приложения на корректность кодов возврата.

**список использованных источников**

1. Docker Docs [Электронный ресурс]. URL: <https://docs.docker.com/> (дата обращения: 28.04.2024)
2. ИС «ИОТ» [Электронный ресурс]. URL: <https://digital.etu.ru/trajectories> (дата обращения 28.04.2024)
3. Linux man pages [Электронный ресурс] URL: <https://linux.die.net/man/> (дата обращения 28.04.2024)

**приложение А**

**Исходный код Проекта**

Исходный код проекта доступен по ссылке:

https://github.com/moevm/devops-1h2024/tree/main/9303\_Shakhin\_NS