**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 9

|  |  |  |
| --- | --- | --- |
| Студент гр. 8346 |  | Товарищев И.В. |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

2023

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Товарищев И.В. | | |
| Группа 8346 | | |
| Тема работы: Использование Docker | | |
| Исходные данные:  Требуется реализовать конфигурацию docker-compose, состоящую из двух контейнеров – с приложением app и с тестами tests. | | |
| Содержание пояснительной записки:  • Содержание  • Введение  • Постановка задачи  • Описание Dockerfile  • Описание скриптов запуска тестов  • Описание конфигурации docker-compose  • Заключение  • Список использованных источников  • Приложение А. Исходный код программы | | |
| Предполагаемый объем пояснительной записки:  Не менее 15 страниц. | | |
| Дата выдачи задания: | | |
| Дата сдачи реферата: | | |
| Дата защиты реферата: | | |
| Студент |  | Товарищев И.В. |
| Преподаватель |  | Заславский М.М. |

СОДЕРЖАНИЕ

|  |  |  |  |
| --- | --- | --- | --- |
| Постановка задачи………………………………………………………. | | | 5 |
| 1. | Описание Dockerfile……………………………………………….. | |  |
|  | 1.1. | Dockerfile для app-контейнера……………………………… | 6 |
|  | 1.2. | Dockerfile для tester-контейнера……………………………. | 6 |
| 2. | Описание скриптов и тестов………………………………………. | |  |
|  | 2.1. | Описание test\_all.sh для запуска тестов……………………. | 7 |
|  | 2.2. | Описание selenium\_test.sh …………….……………………. | 7 |
|  | 2.3. | Описание static\_test.sh …………….……………………….. | 7 |
|  | 2.4. | Описание style\_test.sh …………….……………………….. | 7 |
|  | 2.5. | Описание integration\_test.sh …….……………………….. | 7 |
|  | 2.6. | Статический анализ на проверку собственного критерия.. |  |
|  | 2.7. | Форматирование кода с помощью black………………….. | 7 |
| 3. | Описание конфигурации docker-compose………………………. | | 7 |
| Заключение…….………………………………………………………. | | | 8 |
| Список использованных источников…………………………………. | | | 8 |
| Приложение А. Представление созданных файлов………………….. | | | 9 |

**введение**

В этой работе была реализована docker-compose конфигурация, которая включает в себя два сервиса и соответствующие dockerfile для их сборки. Один контейнер предназначен для запуска веб-сервера, другой предназначен для запуска тестов. Тестирование включает анализ кода при помощи pylint и black, а также интеграционного и selenium тестов.



**Постановка задачи**

Реализовать docker-compose конфигурацию из двух контейнеров:

* app – контейнер для настройки среды;
* tester – контейнер для настройки компонентов тестов и запуском всех тестов.

Для достижения данной задачи написать два файла Dockerfile, которые удовлетворяют следующим требованиям:

* Минимальная версия докера Docker version 19.03.13
* 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
* 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.

В соответствии с вариантом задания выполнить следующие задачи:

* Форматирование Python (black);
* Анализ по 10 существующим критериям;
* Создание интеграционного теста, проверяющего загрузку файла;
* Создание теста используя Selenium, который должен проверять загрузку и получение файлов;
* Добавить внешний доступ по SSH для контейнеров в app и tester - по публичному ключу (существующему);
* Каждый этап тестирования - в docker log (stdout + stderr) + добавить к записям лога timestamp;
* Возможность передачи списка этапов тестирования для запуска с помощью файла .env;
* Ограничить Максимальное Количество процессов.
* Органичения ресурсов - ограничения ресурсов для контейнеров в docker-compose.yml

**ОЗУ** - ограничьте доступную каждому из контейнеров ОЗУ до объема 100 + 9 \* 10 МБ =1000 МБ

**Ядра процессора** - ограничьте доступные в каждом контейнере количество ядер ЦПУ до (1 + НОМЕР\_ВАРИАНТА % 2) (остаток от деления номера вашего варианта на два) = 5

**Максимальное Количество процессов** - ограничьте до количества НОМЕР\_ВАРИАНТА = 9

1. ОПИСАНИЕ Dockerfile
   1. Dockerfile для app-контейнера

Базовый образ для контейнера OS Ubuntu 22.04.

Вначале создания контейнера задается папка выполнения /app. Копируется публичный ключ для доступа по ssh в папку с данными для сервиса. Устанавливаются apt-зависимости и приложения openssh-server python3.

* 1. Dockerfile для tester-контейнера

В качестве базового образа используется Ubuntu 22.04.

Задается папка выполнения /test, копируется содержание папки tests со скриптами в текущую папку. Копируется публичный ключ для доступа по ssh в папку с данными для сервиса. Устанавливаются apt-зависимости и приложения openssh-server python3, xvfb – для запуска firefox без графического интерфейса, libgtk, libdbus и libasound – зависимости, необходимые для запуска firefox. Затем происходит установка приложений, который необходим для проверки с помощью black, pylint и selenium. Далее wget – для скачивания firefox и geckodriver, скачивание и установка firefox и geckodriver, они необходимы для запуска selenium-тестов. В завершении запускается скрипт для запуска всех тестов.

2. ОПИСАНИЕ СКРИПТОВ И ТЕСТОВ

2.1. test\_all.sh для запуска всех проверок.

2.2. static\_test.sh выполняет проверку кода в файле unit\_code\_pylint.py на соответствие стилю кодирования.

2.3. style\_test.sh проверка файла unit\_code\_pylint.py на наличие определенных ошибок (10 шт) определенных в списке вызываемой команды. ! знак в начале командной стоки означает инвертировать наличие ошибок для статуса выполнения.

2.4. selenium\_test.sh запускает приложение selenium и проверяет нахождение и загрузку файла с сайта.

2.5. integration\_test.sh запускает код request.py для проверки загрузки файла.

3. ОПИСАНИЕ КОНФИГУРАЦИИ DOCKER-COMPOSE

Конфигурация docker-compose описывается в файле docker-compose.yml. В нем описано два сервиса – app и tester.

Для запуска сервиса app используется образ из Dockerfile\_app, для tester – образ из Dockerfile\_tester.

Для app задаются следующие параметры:

• Соотнесение портов 5000 на хост-машине и внутри контейнера для доступа к контейнеру по SSH.

• Ограничение на максимальное количество ОЗУ в 1000 МБ.

Для tester задаются следующие параметры:

• Соотнесение порта 5022 на хост-машине с портом 22 внутри контейнера для доступа к контейнеру по SSH.

• Зависимость от сервиса app: для проведения тестирования нужен уже запущенный сервис app.

ЗАКЛЮЧЕНИЕ

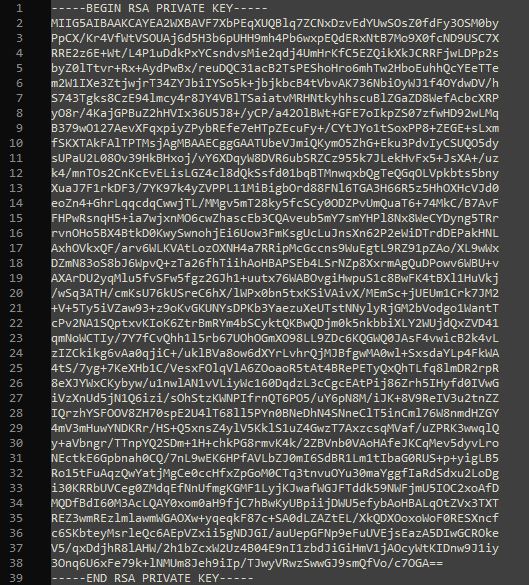
В ходе выполнения данной работы были изучены возможности docker и docker-compose, для этого была реализована конфигурация, состоящая из двух контейнеров – в одном запускается приложение, другой используется для запусков тестов. В ходе работы были решены такие задачи, как запуск firefox в условиях отсутствия графической среды, реализованы различные виды тестов.

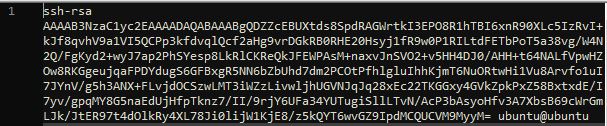
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация по Dockerfile // Docker Docs. URL: <https://docs.docker.com/engine/reference/builder>/ (Дата обращения: 25.04.2023)
2. 2. Документация по файлу конфигурации docker-compose // Docker Docs. URL: <https://docs.docker.com/compose/compose-file>/ (Дата обращения: 28.04.2023)
3. Документация pylint // Pylint User Manual. URL: <https://docs.pylint.org/> (Дата обращения: 21.03.2023)
4. Документация Selenium // The Selenium Browser Automation Project. URL: <https://stepik.org/lesson/228249/step/7> (Дата обращения: 20.03.2023)
5. Документация для использования ssh [Linux handbook](https://linuxhandbook.com/ssh-into-container/) (дата обращения 17.05.2023)

Приложение А. Созданные файлы

id\_rsa (app/id\_rsa, test/id\_rsa)



id\_rsa.pub (app/id\_rsa.pub test/id\_rsa.pub)

Dockerfile\_app

FROM ubuntu:22.04

WORKDIR /app

COPY id\_rsa.pub /root/.ssh/authorized\_keys

RUN apt-get update && apt-get install -y \

    openssh-server=1:8.9p1-3ubuntu0.1 \

    python3-pip=22.0.2+dfsg-1ubuntu0.2 \

    python3=3.10.6-1~22.04

RUN mkdir -p /var/run/sshd

ENTRYPOINT ["echo", "hey!"]

Dockerfile\_tester

FROM ubuntu:22.04

WORKDIR /test

COPY ./tests .

COPY id\_rsa.pub /root/.ssh/authorized\_keys

RUN apt-get update && apt-get install -y \

    openssh-server=1:8.9p1-3ubuntu0.1 \

    python3-pip=22.0.2+dfsg-1ubuntu0.2 \

    python3=3.10.6-1~22.04 \

    git=1:2.34.1-1ubuntu1.9 \

    xvfb=2:21.1.4-2ubuntu1.7~22.04.1 \

    devscripts=2.22.1ubuntu1 \

    libgtk-3-0=3.24.33-1ubuntu2 \

    libdbus-glib-1-2=0.112-2build1 \

    libasound2=1.2.6.1-1ubuntu1 \

    && rm -rf /var/lib/apt/lists/\*

RUN pip install black pylint selenium==4.5.0

RUN mkdir -p /var/run/sshd

RUN wget -O firefox-setup.tar.bz2 "https://download.mozilla.org/?product=firefox-latest&os=linux64" \

    && tar -xvjf firefox-setup.tar.bz2 \

    && ln -s /usr/local/firefox/firefox /usr/bin/firefox \

    && wget https://github.com/mozilla/geckodriver/releases/download/v0.33.0/geckodriver-v0.33.0-linux64.tar.gz \

    && tar -xvzf geckodriver\* \

    && cp geckodriver /usr/bin/geckodriver \

    && chmod 755 /usr/bin/geckodriver

ENTRYPOINT ["./test\_all.sh"]

docker-compose.yml

version: "3"

services:

  app:

    ports:

      - "127.0.0.1:5000:5000"

      - "127.0.0.1:5022:22"

    build:

      context: ./app

      dockerfile: Dockerfile\_app

  tester:

    ports:

      - "127.0.0.1:5021:22"

    build:

      context: ./test

      dockerfile: Dockerfile\_tester

    depends\_on:

      - app

integration\_test.sh

#!/bin/bash

sleep 1

if [ true ]

then

  python3 request.py

  status=$?

  if [ $status -eq 0 ]

  then

    echo "Integration: PASSED"

    exit 0

  else

    echo "Integration: FAILED"

    exit 1

  fi

else

  echo "Integration: DISABLED"

  exit 1

fi

request.py

import shutil

import requests

def discovery ():

  url = 'https://reqbin.com/echo/get/json'

  response = requests.get(url, stream=True)

  with open('sample.json', 'wb') as out\_file:

    shutil.copyfileobj(response.raw, out\_file)

  print('The file was saved successfully')

  return 0

selenium\_test.sh

#!/bin/bash

sleep 1

if [ true ]

then

  python3 selenium\_test.py

  status=$?

  if [ $status -eq 0 ]

  then

    echo "Selenium: PASSED"

    exit 0

  else

    echo "Selenium: FAILED"

    exit 1

  fi

else

  echo "Selenium: DISABLED"

  exit 1

fi

selenium\_test.py

from selenium import webdriver

from unittest import TestCase, main

#class ExampleSeleniumTest(TestCase):

def testSearch(self):

  dirver = webdriver.Firefox()

  dirver.get("https://www.cs/umd.edu/~mount/ANN/")

  driver.implicitly\_wait(30)

  body = driver.find\_element\_by\_link\_text("ann\_1.1.2.zip").click()

  return 0

if \_\_name\_\_ == '\_\_main\_\_':

  main()

static\_test.sh

#!/bin/bash

root=$(dirname $0)

sleep 1

if [ true ]

then

  ! black unit\_code\_pylint.py --check

  status=$?

  if [ $status -eq 0 ]

  then

    echo "Black: PASSED"

    exit 0

  else

    echo "Black: FAILED"

    exit 1

  fi

else

  echo "Black: DISABLED"

  exit 1

fi

style\_test.sh

#!/bin/bash

sleep 1

if [ true ]

then

  ! pylint unit\_code\_pylint.py --enable C0301,E0401,W0401,C0410,E0602,R0903,W0212,R1735,C0209,C0411

  status=$?

  if [ $status -eq 0 ]

  then

    echo "Pylint: PASSED"

    exit 0

  else

    echo "Pylint: FAILED"

    exit 1

  fi

else

  echo "Pylint: DISABLED"

  exit 1

fi

unit\_code\_pylint.py

""" parameter setting of Luc let`s go said a Gibbon when step to new planet.

Lets raise who can introduse wery haigh intelegent think.

We has three examples and so one.

It is programer, scool, driver"""

from PyQt5.QtGui import \*

import aggdraw

import sys, traceback

from PIL import Image, ImageDraw

class WorkerSignals(QObject):

    """Defines the signals"""

    finished = pyqtSignal()

    error = pyqtSignal(tuple)

    progress = pyqtSignal(int)

self.\_data\_lines = dict()

citem, vitem = self.get\_or\_create\_data\_row(currency)

vitem.setText("%.4f" % data["close"])

img = Image.new("RGB", (300, 300), (255, 255, 255))

draw = ImageDraw.Draw(img)

draw.ellipse((0, 0, 150, 150), fill="red", outline="red")

pen = aggdraw.Pen("red", 0.5)

brush = aggdraw.Brush("red")

draw2 = aggdraw.Draw(img)

draw2.ellipse((150, 150, 300, 300), pen, brush)

draw2.flush()

img.show()

try:

    x = 1 / 0

except ZeroDivisionError:

    Type, Value, Trace = sys.exc\_info()

    print ("Type: ", Type)

    print ("Value:", Value)

    print ("Trace:", Trace)

    print ("\n", "print\_exception()".center(40, "-"))

    traceback.print\_exception(Type, Value, Trace, limit=5,file=sys.stdout)

    print ("\n", "print\_tb()".center(40, "-"))

    traceback.print\_tb(Trace, limit=1, file=sys.stdout)

test\_all.sh

#!/bin/sh

root=$(dirname $0)

echo "\033[31m===============\nRun pipeline\n===============\033[0m"

start\_black\_linter() {

    echo "\n\033[35m=============== Run black linter test ===============\033[0m\n"

    bash ./static\_test.sh

    static=$(echo $?)

}

start\_pylint() {

    echo "\n\033[35m===============Run PyLint for 10 categories test===============\033[0m\n"

    bash ./style\_test.sh

    style=$(echo $?)

}

start\_selenium\_tests() {

    echo "\n\033[35m=============== Run Selenium tests ===============\033[0m\n"

    bash ./selenium\_test.sh

    selenium=$(echo $?)

}

start\_integration\_tests() {

    echo "\n\033[35m=============== Run Integration tests ===============\033[0m\n"

    bash ./integration\_test.sh

    integration=$(echo $?)

}

get\_report() {

  echo "\033[031m===============\n Results \n===============\033[0m"

  if [ "$style" -eq 0 ]

  then

      echo "\033[32mStyle: PASSED\033[0m"

  else

      echo "\033[31mStyle: FAILED\033[0m"

  fi

  if [ "$static" -eq 0 ]

  then

      echo "\033[32mStatic: PASSED\033[0m"

  else

      echo "\033[31mStatic: FAILED\033[0m"

  fi

  if [ "$selenium" -eq 0 ]

  then

      echo "\033[32mSelenium: PASSED\033[0m"

  else

      echo "\033[31mSelenium: FAILED\033[0m"

  fi

  if [ "$integration" -eq 0 ]

  then

      echo "\033[32mIntegration: PASSED\033[0m"

  else

      echo "\033[31mIntegration: FAILED\033[0m"

  fi

}

case "$1" in

  -p)

        start\_pylint

        ;;

  -h)

        start\_black\_linter

        ;;

  -s)

        start\_selenium\_tests

        ;;

  -i)

        start\_integration\_tests

        ;;

  \*)

        echo "\n\033[31m=============== Run all tests ===============\033[0m\n"

        start\_black\_linter

        start\_pylint

        start\_integration\_tests

        start\_selenium\_tests

        echo "\n\033[31m=============== End all tests ===============\033[0m\n"

        get\_report

        ;;

esac