ЛАБОРАТОРНА РОБОТА № 1

Дослідження системи контейнерів Docker

Мета роботи: полягає у дослідженні специфіки запуску Docker контейнерів, ознайомленні з репозиторієм Docker Hub та, за потреби, Docker Desktop.

Вхідні дані ЛР1

У якості вхідних даних для ЛР1 є:

- за використання Windows: встановлений Docker Desktop або віртуальну машину Linux зі встановленим Docker;
- за використання Linux: встановлений Docker.

Додатково слід пам’ятати що в цій лабораторній роботі:

* – без символів “<>”;

** – прізвища AAA=1+1+1=3=8003, а ZYXZ=26+25+24+26=101=8101.

Вихідні дані ЛР1

У якості вихідних даних для ЛР3 є: робочі контейнери та звіт.

Завдання

1. Навчитися піднімати контейнери Docker.
2. Навчитися працювати з вебсервером nginx.

Програма проведення експерименту ЛР 1

1. На базі alpine/nginx створити власний image NAME=lab01_1br<перші літери прізвищ кожного з членів бригади>*. Команди і результат роботи системи показаний на рис.1.

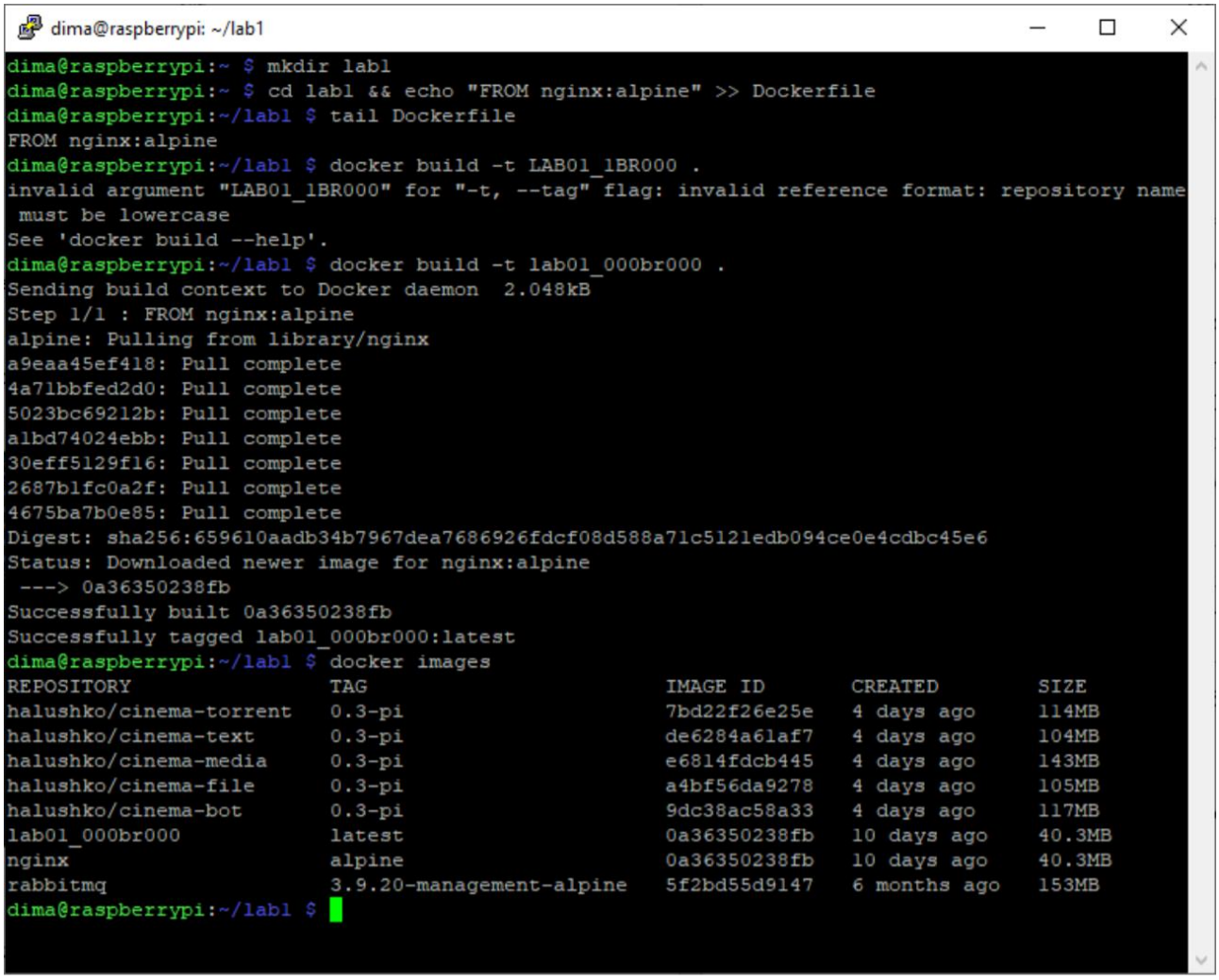


Рисунок 1 – Створення власного образу

2. Користуючись командою docker images, вивести перелік образів (images). Результат зафіксувати у вигляді скриншоту.

3. Запустити контейнер, вивести інформацію по активним контейнерам. Приклад команд і результати роботи показані на рис.2. На цьому рисунку показані приклад запуску контейнеру з лабораторної роботи і результат її виконання.

Після отримання результату, потрібно зафіксувати його у вигляді скриншоту. Результат додати до звіту.

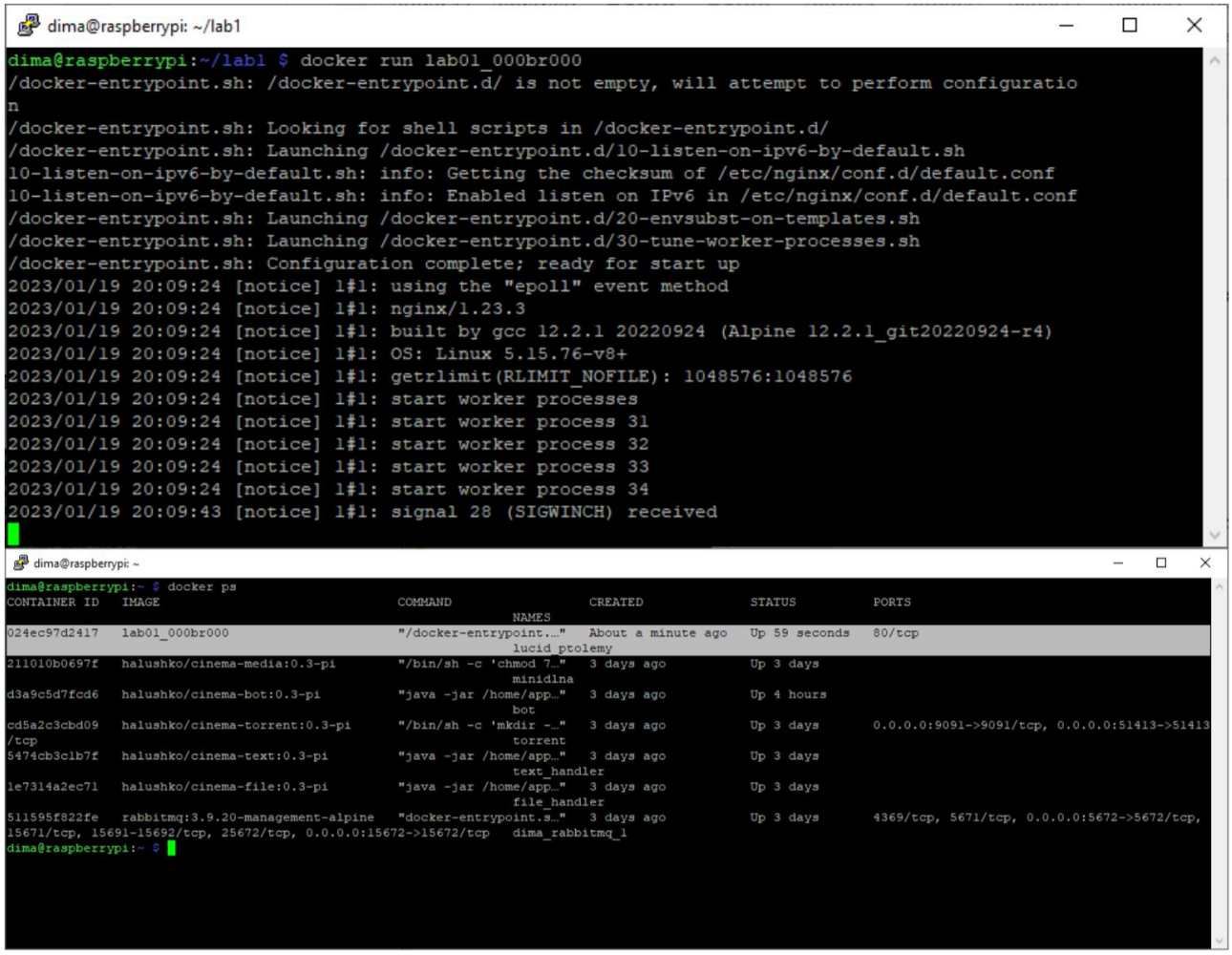


Рисунок 2 – Приклад запуску контейнерів

4. Зупинити контейнер, користуючись командою, що показана на рис.3. Вивести інформацію по активним контейнерам. Після отримання результату, потрібно зафіксувати його у вигляді скриншоту. Результат додати до звіту.

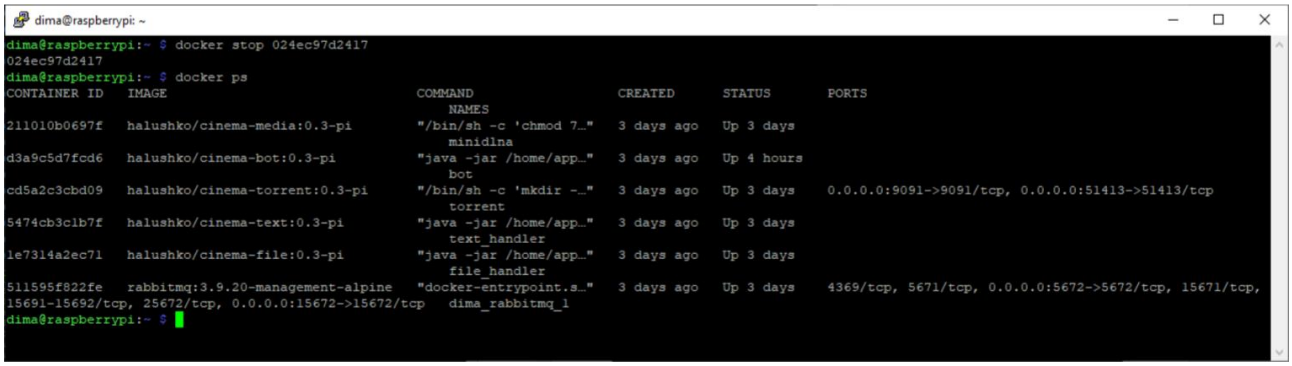


Рисунок 3 – Приклад зупинки контейнеру

5. Вивести інформацію по усім контейнерам (ключ -a). Після отримання результату, потрібно зафіксувати його у вигляді скриншоту. Результат додати до звіту.

6. Сформувати звіт.

Контрольні запитання до лабораторної роботи № 1

1. Docker – основні поняття та команди.

2. Поясніть у чому відмінність між Docker IMAGE та Docker CONTAINER?
3. Що означає тег ALPINE в докер образах?
4. Яким чином можна переглянути контейнери, що були зупинені?
5. Яка роль Dockerfile при створенні образів?

ЛАБОРАТОРНА РОБОТА № 2

Дослідження спільних ресурсів хостової та гостьової систем в Docker

Мета роботи: полягає у дослідженні специфіки запуску Docker контейнерів, ознайомленні з репозиторієм Docker Hub та, за потреби, Docker Desktop. Навчитися прокидати порти з гостьової на хостову машини, що дасть змогу працювати з власним веб-сервером nginx.

Вхідні дані ЛР2

У якості вхідних даних для ЛР2 є:

– виконана ЛР1 та її Docker-образи.

Додатково слід пам’ятати що в цій лабораторній роботі:

* – без символів “<>”;

** – прізвища AAA=1+1+1=3=8003, а ZYXZ=26+25+24+26=101=8101.

Вихідні дані ЛР2

У якості вихідних даних для ЛР2 є: робочий контейнер з веб-сервером nginx, звіт.

Завдання

1. Навчитися використовувати спільні ресурси хостової та гостьової систем на прикладі Docker nginx.
2. Переглянути контет сторінок Docker nginx з хостової машини.

Програма проведення експерименту ЛР2

1. Прокинемо порти. В команду docker run додати такі ключі, щоб на сайт контейнера можна було зайти через 127.0.0.1:<80 + двозначна сума номерів перших літер ваших прізвищ>**. Вивести інформацію по активним контейнерам. Приклад позитивного виконання показаний на рис.4. Після отримання результату, потрібно зафіксувати його у вигляді скриншоту. Результат додати до звіту.

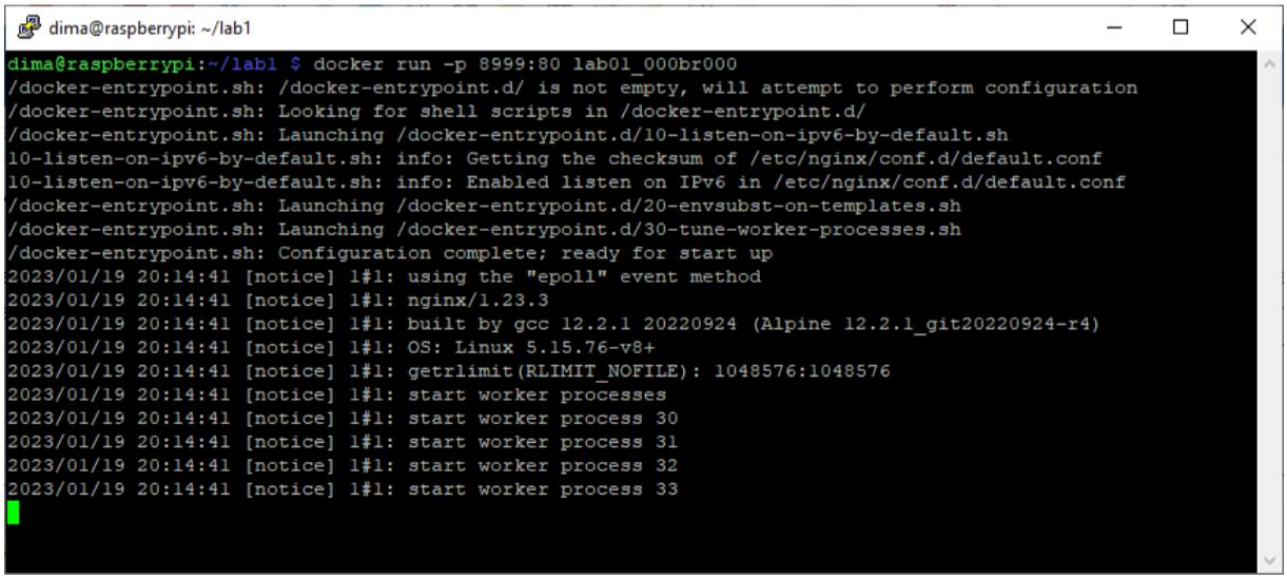


Рисунок 4 – Запуск контейнера з прокиданням портів.

Для перевірки результату через браузер гостьової машини потрібно звернутися до відповідної адреси і порту, а саме перейти на 127.0.0.1:<порт>; у браузері, як це показано на рис. 5. Після отримання результату, потрібно зафіксувати його у вигляді скриншоту. Результат додати до звіту.

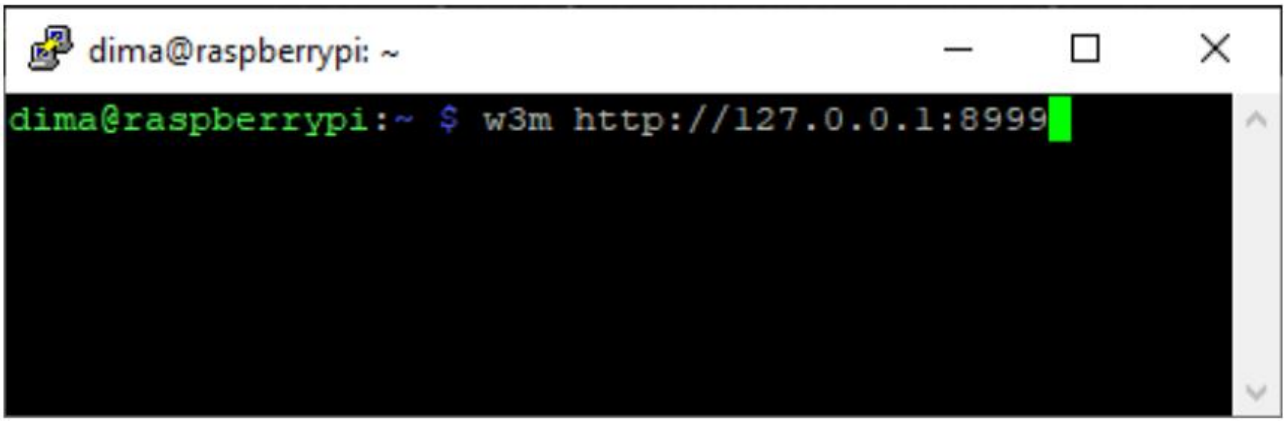


Рисунок 5 – Приклад запиту

2. Замінити контент дефолтної сторінки nginx на власний – перелік ПІБ всіх членів бригади + поточна дата створення image. Замінити через ADD/COPY та, за потреби RUN, у Dockerfile. Створити НОВИЙ image NAME=lab01_2b<перші літери прізвищ кожного з членів бригади> на базі імейджу lab01_1brxxx. Зайти на сторінку через браузер. Приклад показаний на рис.6.

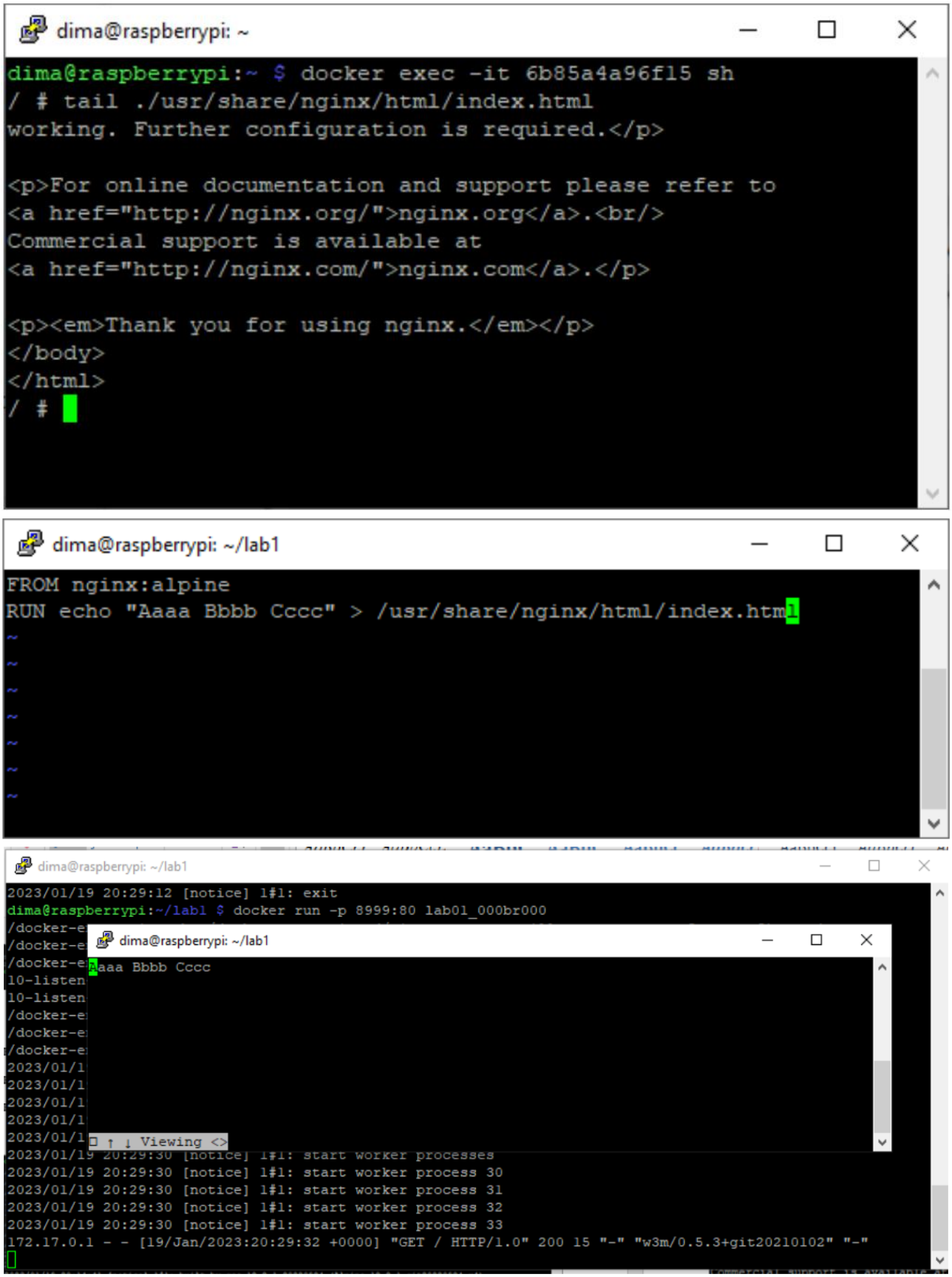


Рисунок 6 – Заміна контенту дефолтної сторінки

3. Замінити контент дефолтної сторінки nginx на власний, але розшарений з каталогу ./lab01 – односторінковий WEB-застосунок, який презентуватиме склад бригади (з вказанням хобі, тощо), який але створіть файл у папці ./lab01 і зробити так, щоб через VOLUME цей файл ставав сторінкою. Створити НОВИЙ

image NAME=lab01_3brxxx на базі імейджу lab01_1brxxx. Зайти на сторінку через браузер. Після отримання результату, потрібно зафіксувати його у вигляді скриншоту. Результат додати до звіту.
4. Сформувати звіт.

Контрольні запитання до лабораторної роботи № 2
1. Docker – інструкції Dockerfile.
2. Що таке прокидання портів та навіщо його робити при запуску докер контейнерів?
3. Що таке VOLUME у докері?
4. Що виконує інструкція RUN у Dockerfile?
5. Яка команда дозволяє зайти в командний рядок контейнера Docker?

ЛАБОРАТОРНА РОБОТА № 3

Створення та керування портативними віртуальними середовищами

Мета роботи: полягає у дослідженні специфіки створення та налаштування віртуальних машин за допомогою Vagrant, ознайомленні з репозиторієм Vagrant Cloud. Навчитися прокидати порти з гостьової на хостову машини, що дасть змогу працювати з власним веб-сервером nginx. Закріпити на практиці різницю між контейнеризацією та віртуалізацією.

Вхідні дані ЛР3

У якості вхідних даних для ЛР3 є:
- web-застосунок з ЛР2, який представляє склад бригади.

Додатково слід пам’ятати що в цій лабораторній роботі:

* – без символів “<>”;

** – прізвища AAA=1+1+1=3=8003, а ZYXZ=26+25+24+26=101=8101.

Вихідні дані ЛР3

У якості вихідних даних для ЛР3 є: піднята віртуальна машина з веб-сервером nginx, звіт.

Завдання

1. Навчитися створювати на налаштовувати віртуальне середовище.
2. Навчитися прокидати порти з гостьової на хостову машини.
3. Переглянути контет сторінок nginx з хостової машини.

Програма проведення експерименту ЛР3

1. Після інсталяції Vagrant, обрати/створити папку для виконання лабораторної роботи та створити робоче оточення (ініціалізувати Vagrant –

виконується даний крок за допомогою команди «vagrant init»). Скриншот процесу додати до звіту.

2. Встановити nginx на створену віртуальну машину («config.vm.hostname» якої має складатись з перших літер прізвищ кожного з членів бригади). Тут починається множинність різних підходів, наприклад: виконати це можна вручну після підключення по SSH, автоматично після створення віртуальної машини за допомогою написаного bash-скрипта (вказується у Vagrant файлі), або навіть за допомогою Ansible (згадується в контрольних питаннях, але ретельно досліджувати інструменти SCM потреби немає – ознайомтесь хоча б з їх призначенням). Паралельно виконайте прокидання портів за тим же принципом, що і в Лабораторній роботі №2: :<80 + двозначна сума номерів перших літер ваших прізвищ>** – налаштування «config.vm.network» у Vagrant файлі. Для перевірки результату потрібно через браузер гостьової машини звернутися до відповідної адреси і порту.

Після отримання результату, потрібно зафіксувати його у вигляді скриншоту, те саме стоється Vagrant файлу та обраного процесу встановлення nginx. Додати все до звіту.

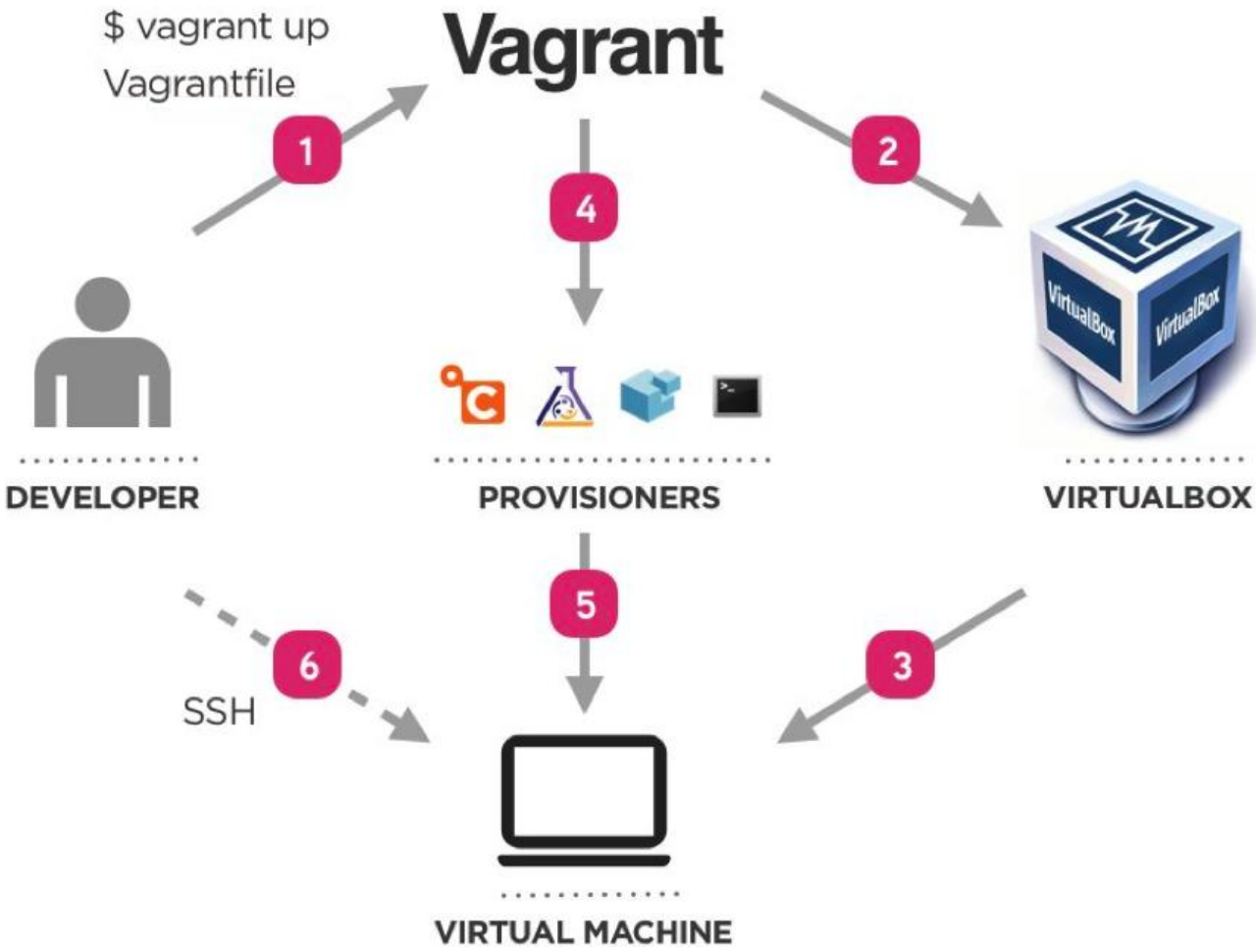
3. Замінити контент дефолтної сторінки nginx на власний – використати свій же застосунок з Лабораторної роботи №2. Знову ж можна виконати цей крок декількома способами. Результат в браузері та обраний процес заміни знову ж таки зафіксувати у вигляді скриншоту та додати все до звіту.

(3.5) НЕ обов’язкове, але може бути цікавим для тих, хто саме «ручками» встановлював nginx та замінював дефолтну сторінку – поцікавтесь як автоматизувати даний процес (перевірка вмісту дефолтної сторінки, звісно ж, відпадає). Можете теж додати до звіту.

4. Сформувати звіт.

Контрольні запитання до лабораторної роботи № 3

1. Поняття та приклади провайдерів віртульних машин.
2.Vagrant Boxes та Vagrant Cloud.
3. Vagrantfile – ініціалізація та інструкції для налаштувань.
4. Команди Vagrant.
5. Розібратись що відбувається на картинці нижче та познайомитись з поняттям SCM (System Configuration Management): Ansible, Chef, Puppet і про існування Terraform теж дізнатись (без вдавання в деталі)?



ЛАБОРАТОРНА РОБОТА № 4

Створення і розгортання програмної інфраструктури на основі docker-compose

Мета роботи: полягає у дослідженні процесу автоматичного розгортання відносно складної програмної інфраструктури розподіленого веб-застосунку за обраним напрямом технології. Зважаючи на те, що сучасні РПС являють собою систему програмних модулів, що взаємодіють між собою і реалізовані на різних технологіях, їх автоматичне розгортання потребує додаткових програмних механізмів, що спрощують процес розгортання і розробки.

Одним з таких механізмів є docker-compose. У цій ЛР відбувається розгортання РПС з різних модулів відповідно до наданого завдання, вивчення синтаксисту файлів docker-compose і тестування отриманих результатів.

Лабораторну роботу можна умовно розподілити на три частини:

- вивчення і тестування складових частин відносно складної РПС, дозволяється використовувати власні напрацювання і досвід роботи за фахом студентів, що навчаються;

- підготовка первинного файлу docker-compose, вивчення елементів синтаксису і формування стилю цього файлу;

- розгортання РПС з використанням отриманого docker-compose, тестування роботи складної програмної системи, виправлення помилок, що були виявлені.

Вхідні дані ЛР4

У якості вхідних даних для ЛР4 є:

- 2 бази даних (SQL та NoSQL), які містяться у відповідних стандартних контейнерах, які можуть бути активовані з docker-compose;
- два або три контейнера з базовим веб-застосунком, що побудовані на основі типового веб-фреймворку і взаємодіють з базами даних.

Вихідні дані ЛР4

У якості вихідних даних для ЛР4 є: система каталогів з файлом docker-compose.yml, звіт.

Завдання

1. У якості індивідуального завдання, на першому етапі слід вивчити переваги і недоліки баз даних (є надані у переліку варіантів, але ж дозволяються і власні варіанти), за необхідності побудувати файли docker-compose для баз даних і протестувати роботу цього файлу на реальній системі, використовуючи доступні засоби тестування БД.

Вивчити веб-застосунок, що побудований на основі фреймворку з наданих варіантів модифікувати його для роботи з вашими БД, або одразу працювати з обраним самостійно варіантом. При необхідності провести його дослідження з docker-compose, аналогічно БД.

2. Підготувати файл docker-compose, що дозволяє побудувати узагальнену систему з парою контейнерів з БД і двома-трьома контейнерами з веб-застосунку (для front/back end), які взаємодіють відповідно до функціоналу з БД.

3. Провести розгортання РПС з використанням отриманого docker-compose. Продумати послідовність запуску контейнерів. Після розгортання провести тестування роботи складної програмної системи, що запущена з використанням результатів п.2. У випадку наявності, провести виправлення помилок. При неможливості поєднати БД і фреймворк веб-застосунку, провести аналіз помилок та обрати іншу БД або фреймворк.

Стек технологій, обирається студентами за власними вподобаннями.

Для SQL бази даних можна обрати postgres:14.1-alpine, mysql або іншу.

Для NoSQL БД гарно підійде MongoDB, але вибір теж не обмежений.

У якості фреймворку можна обрати Django або Spring, хоча можна створити застосунок і на свій смак.

Варіанти завдання

Можна придумати завдання самостійно, але головне, що для всіх таблиць (можна навіть обійтись лише двома, але обов'язково SQL/NoSQL) мають виконуватись CRUD операції:

Варіант №1. Запити агентства з реклами.

Таблиці: Клієнти (Код, ПІБ, Вік, Стать + можливо ще щось) – для NoSQL. Співробітники (Код, ПІБ, Вік, Стать). Співробітники і посади (Код, посади, Оклад, Обов'язки, Вимоги).

Відділ кадрів (Зв'язує таблиці «Співробітники» і «Співробітники і посади» по полю «Код посади»).

Фільтри: Фільтр для відображення співробітників окремих посад (на основі запиту «Відділ кадрів»); клієнтів.

Варіант №2. Фірма з продажу персональних комп'ютерів.

Таблиці: Клієнти (Код, ПІБ, Вік, Стать, тип, вартість). Товар (Тип персональних комп'ютерів, вартість). Продавці (Код, ПІБ, Вік, Стать + можливо ще щось) – для NoSQL.

Замовлення (Зв'язує таблиці «Клієнти» і «Товар» у контексті замовлення).

Фільтр: Фільтр для відображення клієнтів та замовлень; продавців

Варіант №3. Продаж автомобілів

Таблиці: Клієнти (Код, ПІБ, Вік, Стать, тип автомобілю, вартість). Автомобілі (Код, тип автомобілю, вартість, пробіг, технічний стан). Продавці (Код, ПІБ, Вік, Стать + можливо ще щось) – для NoSQL.

Замовлення (Зв'язує таблиці «Клієнти» і «Автомобілі» у контексті замовлення).

Фільтр: Фільтр для відображення клієнтів, замовлень; продавців.

Варіант №4. Кінотеатр

Таблиці: Глядач (Код, ПІБ, Вік, Стать, замовлення). Квиток (Код, дата, номер місця, час, назва фільму). Продавці (Код, ПІБ, Вік, Стать + можливо ще щось) – для NoSQL.

Замовлення (Зв'язує Таблиці «Глядач» і «Квиток» у контексті замовлення).

Фільтр: Фільтр для відображення замовлень по певним датам; продавців.

Програма проведення експерименту ЛР4

Залежно від обраних технологій та завдання хід проведення експерименту може відрізнятись, тому орієнтовні програма наступна.

1. Перший етап.

Перед початком досліджень перевіряємо працездатність Docker compose для обраного типу операційної системи. Для деяких ОС засіб Docker compose постачається за замовчуванням разом з Docker, для деяких ОС потрібно його додатково інсталювати. При необхідності провести додаткову інсталяцію з використанням стандартних засобів, для обраної операційної системи. Створюємо робочий каталог проекту. У робочий каталог проекту розміщуємо файл docker-compose.yml.

2. Другий етап.

Користуючись мануалом [4 д.], створюємо два мінімальних проекти Django у робочому каталозі проекту (РКП) разом з веб-застосунком на базі типового мануала. Достатньо виконати простий проект на рівні двох частин мануалу [4 д.] «Writing your first Django app, part 1», «Writing your first Django app, part 2».

Результатом мають бути два внутрішніх каталоги у РКП з проектами Django. Провести тестування двох проектів на працездатність. Як це зробити показано у мануалі [4 д.] частина 1.

Користуючись мануалом [4 д.] переналаштувати отримані контейнери на роботу з базами даних.

3. Третій етап.

Для кожного проекту у РКП слід додати Dockerfile з відповідними налаштуваннями, з урахуванням БД. Виконати всі потрібні дії для підготовки системи контейнерів.

4. Четвертий етап.

Відповідно до функціоналу слід внести зміни до файлу docker-compose.yml. Передбачено, що будуть працювати чотири контейнера. Два контейнера з проектом Django і два контейнера з базами даних. Після внесення змін до файлу docker-compose.yml перевіряємо результат docker-compose build. При успішній побудові доступним є тест двох проектів Django. У випадку помилок, внести відповідні зміни до файлів docker-compose.yml і провести додаткове тестування.

5. П’ятий етап.

Далі налаштовуємо БД. Для цього вносимо відповідні зміни до файлу docker-compose.yml. Оновлюємо налаштування у двох проектах Django налаштовуємо додаткові елементи. Після внесених змін проводимо тестування результатів. Після отримання результату, потрібно зафіксувати його у вигляді скриншоту. Результат додати до звіту.

6. Сформувати звіт.

Контрольні запитання до лабораторної роботи № 4

1. Поясніть у чому відмінність технології Docker і Docker compose?
2. Які етапи створення складної розподіленої програмної системи з використанням технології Docker compose?
3. Розкрийте основу структури файлу docker-compose.yml.
4. Інструкції Docker compose.
5. Який зв'язок файлу docker-compose.yml і dockerfile?
6. Розкрийте і опішить структуру складного розподіленого проекту, що побудований за технологією Docker compose?
7. Які команди подаються для розгортання складного розподіленого проекту та як забезпечити правильний порядок розгортання його компонентів з використанням Docker compose?
8. Наведіть порядок створення складного розподіленого проекту з використанням Docker compose?

СПИСОК ЛІТЕРАТУРИ

Базова література:

1. Проектування інформаційних систем: Загальні питання теорії проектування ІС (конспект лекцій) [Електронний ресурс]: навч. посіб. для студ. спеціальності 122 «Комп'ютерні науки» / КПІ ім. Ігоря Сікорського; уклад.: О. С. Коваленко, Л. М. Добровська. – Електронні текстові дані (1 файл: 2,02 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2020. – 192с. https://ela.kpi.ua/bitstream/123456789/33651/1/PIS_KL.pdf
2. Інфраструктура програмного забезпечення WEB-застосувань. Лабораторний практикум [Електронний ресурс]: навч. посіб. для студ. спеціальностей 121 «Інженерія програмного забезпечення», 126 «Інформаційні системи та технології» / КПІ ім. Ігоря Сікорського; автор.: М.М. Букасов, Д.О. Галушко, П.Ю. Катін, Я.В. Хіцко. – Електронні текстові дані (1 файл: 1,1 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2023. – 53 с. https://ela.kpi.ua/bitstream/123456789/53028/1/IPZ_WEB-zastosuvan_LP.pdf

Додаткова література:

1. ALAN DENNIS, BARBARA HALEY WIXOM, ROBERTA M. ROTH. System Analisis amd desigm.Fifth Edition. John Wiley & Sons, Inc.2012. 563 c.
2. Django документація [Електронний ресурс] – https://docs.djangoproject.com/en/3.2/.
3. Adam Freeman. Essential Docker for ASP.NET Core MVC.ISBN-13 (pbk): 978-1-4842-2777-0 ISBN-13 (electronic): 978-1-4842-2778-7.2017p.
4. https://docs.djangoproject.com/en/4.1/intro/tutorial01/
5. https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-compose-on-ubuntu-20-04

6. Rebeka Mukherjee. Python Scripting for System Administration. Department of Computer Science and Engineering Netaji Subhash Engineering College, Kolkata.
7. https://docs.docker.com/samples/django/
8. https://docs.docker.com/compose/reference/
9. Adam Freeman. Pro ASP.NET Core 3: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages.2020.
10. Docker документація [Електронний ресурс] – https://docs.docker.com/get-started/.
11. Matthes E. Python Crash Course (2nd Edition) : A Hands-On, Project-Based Introduction to Programming / Eric Matthes. – San Francisco, United States: No Starch. Press, US, 9. – 544 c. – (2nd Edition).
12. Thomas D. The Pragmatic Programmer : your journey to mastery, 20th Anniversary Edition / D. Thomas, A. Hunt. – Boston, United States: Pearson Education. (US), 2020. – 352 c.
13. Learn Python the Hard Way : A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code – New Jersey, United States: Pearson Education. (US), 2013. – 320 c.
14. Learning React : Modern Patterns for Developing React Apps – Sebastopol, United States: O'Reilly Media, Inc, USA, 2020. – 300 c.
15. Docker : Complete Guide To Docker For Beginners And Intermediates, 2020. – 140 c.
16. Docker: Up & Running : Shipping Reliable Containers in Production – Sebastopol, United States: O'Reilly Media, Inc, USA, 2018. – 347 c.
17. Docker homepage - http://www.docker.com/
18. Docker Hub - https://hub.docker.com
19. Docker blog - http://blog.docker.com/
20. Docker documentation - http://docs.docker.com/
21. Docker Getting Started Guide - http://www.docker.com/gettingstarted/
22. Docker code on GitHub - https://github.com/docker/docker
23. Docker on Twitter - http://twitter.com/docker
24. Get Docker help on Stack Overflow - http://stackoverflow.com/search?q=docker
25. Valeria Cardellini. Matteo Nardelli. Container-based virtualization: Docker. Università degli Studi di Roma "Tor Vergata" Dipartimento di Ingegneria Civile e Ingegneria Informatica Corso di Sistemi Distribuiti e Cloud Computing A.A. 2017/18.
26. Adam Freeman. Pro Angular 6 .ISBN-13 (pbk): 978-1-4842-3648-2 ISBN-13 (electronic): 978-1-4842-3649-9/2018 p.