**Міністерство освіти і науки України**
**Тернопільський національний технічний університет імені Івана Пулюя**

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету )
Кафедра комп'ютерних наук
(повна назва кафедри)

# КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Інформаційна юридична система на основі Docker-контейнера

Виконав: студент IV курсу, групи СТс-42
спеціальності 126 Інформаційні системи та технології
(шифр і назва спеціальності)

_____ Ожанський С.І.
(підпис) (прізвище та ініціали)

Керівник _____ Липак Г.І.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Михайлишин М.С.
(підпис) (прізвище та ініціали)

Тернопіль - 2021

Міністерство освіти і науки України
**Тернопільський національний технічний університет імені Івана Пулюя**

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2021 р.

# ЗАВДАННЯ
## НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 126 Інформаційні системи та технології
(шифр і назва спеціальності)

Студенту Ожанському Степану Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна юридична система на основі Docker-контейнера

Керівник роботи Липак Галина Ігорівна, к.т.н., доц. каф. КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «01» 02 2021 року № 4/7-63

2. Термін подання студентом завершеної роботи 26.06.2021р.

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз досліджуваної області. 1.1 Історія концепції віртуалізації. 1.2. Віртуалізація на основі віртуальних машин. 1.3. Контейнерна віртуалізація. 2. Проектна частина 2.1 Програмні технології для контейнерної віртуалізації. 2.2. Порівняння віртуальної машини з контейнерною віртуалізацією. 2.3. Технологія Docker. 2.4. Технології, які застосовує Docker 2.5. Інсталяція Docker і Docker-репозиторія. 3. Програмна реалізація додатка на прикладі модуля відображення фактів. 3.1. Опис модуля відображення фактів. 3.2. Створення образу контейнера. 3.3. Налаштування системи автоматичного складання і публікації образів. 3.4. Порівняння тимчасових метрик після застосування технології Docker 4. Безпека життєдіяльності, основи хорони праці. Висновки. Перелік використаних джерел

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титулка. 2. Актуальність. 3. Мета і задачі дослідження. 4. Технології віртуалізації 5. Контейнерна віртуалізація. 6. Порівняння віртуальної машини з контейнерною віртуалізацією. 7. Технологія Docker. 8. Архітектура Docker, діаграма звернень. 9. Інсталяція Docker. 10. ПЗ, яке використано в роботі. 12. Скріншоти і лістинги. 13. Порівняння тимчасових метрик після застосування технології Docker. 14. Висновки по роботі.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Безпека життєдіяльності, основи хорони праці | Гурик О.Я., доцент кафедри МТ | | |

7. Дата видачі завдання 2021 р.

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Ознайомлення з завданням до кваліфікаційної роботи | 01.02 – 05.02 | *Виконано* |
| 2. | Підбір джерел про процес віртуалізації | 05.02 – 15.02 | *Виконано* |
| 3. | Опрацювання джерел про особливості віртуалізації | | |
| | на основі віртуальних машин та контейнерну | | |
| | віртуалізацію | 16.02 – 27.02 | *Виконано* |
| 4. | Виконання дослідження щодо інформаційної системи на | | |
| | основі Docker-контейнера | 28.02 – 15.03 | *Виконано* |
| 5 | Розроблення програмного коду | 16.03 – 10.04 | *Виконано* |
| 6. | Оформлення розділу «Аналіз досліджуваної області» | 11.04 – 20.04 | *Виконано* |
| 7. | Оформлення розділу «Проектна частина» | 21.04 – 29.04 | *Виконано* |
| 8. | Оформлення розділу «Програмна реалізація додатка на | | |
| | прикладі модуля відображення фактів» | 30.04 – 11.05 | *Виконано* |
| 9. | Виконання завдання до підрозділу «Безпека | | |
| | життєдіяльності, основи хорони праці» | 12.05 – 19.05 | *Виконано* |
| 10. | Оформлення кваліфікаційної роботи | 20.05 – 28.05 | *Виконано* |
| 11. | Нормоконтроль | 05.06 – 12.06 | *Виконано* |
| 12. | Перевірка на плагіат | 12.06 – 15.06 | *Виконано* |
| 13. | Попередній захист кваліфікаційної роботи | 16.06 – 19.06 | *Виконано* |
| 14. | Захист кваліфікаційної роботи | 26.06 | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Студент _____ Ожанський С.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Липак Г.І.
(підпис) (прізвище та ініціали)

# АНОТАЦІЯ

Інформаційна юридична система на основі Docker-контейнера // Кваліфікаційна робота бакалавра // Ожанський Степан Ігорович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, кафедра комп'ютерних наук, група СТс–42 // Тернопіль, 2021 // С. – 52 , рис. – 23, табл. – 1, слайдів – 13 , бібліогр. – 18.

Ключові слова: DOCKER, ГІПЕРВІЗОР, ВІРТУАЛІЗАЦІЯ, КОНТЕЙНЕР, ОБРАЗ

Кваліфікаційна робота присвячена дослідженню сучасних технологій віртуалізації та розробці програмного додатку юридичної інформаційної системи з метою вирішення проблеми поновлення програмного забезпечення, яка полягає в швидкості його первісного встановлення, постачання оновлень, їх застосування, а також в швидкості відновлення функціонування системи у разі збоїв.

Досліджено і порівняно платформи контейнерної віртуалізації (OpenVZ, LXC і Docker). Розроблено вдосконалений підхід і технологічний процес постачання, встановлення і запуску множини екземплярів клієнт-серверного додатка для роботи юридичної інформаційної системи за допомогою Docker-контейнерів шляхом модифікації додатка і його адаптації до даного підходу.

Це дозволило скоротити принаймні вдвічі витрати часу на процеси інсталяції і оновлення модулів інформаційної юридичної системи.

# ANNOTATION

Information legal system based on Docker -container // Ozhanskiy Stepan // Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science // Ternopil, 2021 // P. - 52, Fig. - 23, Table - 1, Slide - 13 , References - 18.

Keywords: DOCKER, HYPERVISOR, VIRTUALIZATION, CONTAINER, IMAGE

Thesis deals with the study of modern virtualization technologies and software development of law information system to solve the problem of software update, which is the speed of its initial installation, supply of updates, their application, as well as the speed of system recovery in case of failure.

Container virtualization platforms (OpenVZ, LXC and Docker) have been studied and compared. An improved approach and technological process of delivery, installation and launch of multiple instances of the client-server application for the law information system using Docker-containers by modifying the application and adapting it to this approach.

This has reduced at least half the time spent on the installation and upgrade of legal information system modules.

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ СКОРОЧЕНЬ І ТЕРМІНІВ

АЗ – апаратне забезпечення.

БД – база даних.

ВМ – віртуальна машина.

ВЗ – віртуалізація

ГВ - гіпервізор

ІС – інформаційна система.

ІТ – інформаційні технології.

ОС – операційна система.

Підхід REST (Representational State Transfer - Передача станів уявлень) – стиль архітектури серверного ПЗ, який використовується для розробки веб-сервісів. В архітектурі REST кожна окрема одиниця інформації однозначно визначається за допомогою URL (ідентифікатор в мережі – UniformResourceLocator).

ПЗ- програмне забезпечення.

Репозиторій- сховище, місце, де зберігаються будь-які дані (частіше у вигляді файлів).

СУБД – система управління базами даних.

ФС – файлова система.

Хостова ОС - ОС, яка встановлена на реальне АЗ і використовується як носій примірників віртуальних ІС, з допомогою установленого ПЗ для реалізації ВЗ.

API (Application Programming Interface) – програмний інтерфейс до датку; набір готових класів, процедур, функцій, структур і констант, наданих додатком для використання у зовнішніх програмних продуктах.

RESTful API - програмний інтерфейс (API), який задовольняє архітектурі REST.

HTTP (Hypertext Transfer Protocol - протокол передачі гіпертексту) - протокол передачі даних.

JSON (JavaScriptObjectNotation) - текстовий формат обміну даними між інформаційними системами, заснований на об'єктах мови програмування JavaScript.

MySQL - вільно розповсюджувана реляційна СУБД.

PVM (Parallel Virtual Machine) - програмний засіб для забезпечення роботи паралельної мережі комп’ютерів.

UnionFS (Union File Systems) – єдина файлова система.

# ЗМІСТ

# ВСТУП

В даний час діяльність практично будь-якого роду пов'язана з інтенсивним застосуванням ІТ, а гасло «хто володіє інформацією, той володіє світом» актуальне, мабуть, як ніколи. Природно, що разом із зростанням потреби в пов'язаних з ІТ послуги у фірм, зростає і потреба в технологіях, які допомагають задовольняти ці запити. Наприклад, організовуються потужні інформаційні центри для хостингу і контролю великих потоків мережевого трафіку, на великі відстані прокладаються канали широкосмугової передачі даних. Також зростає потреба у своєчасному оновленні ПЗ з метою його поліпшення, оптимізації або налагодження. Від того наскільки своєчасно і швидко оновлюються ІС залежить задоволеність кінцевого користувача даних систем.

В даний час досить велика частина клієнт-серверного ПЗ функціонує на стороні замовника, безпосередньо того, хто отримує вигоду від цього ПЗ. Відповідно це накладає свої обмеження на швидкість встановлення і оновлення всієї системи, а також на якість підтримки. Також все серверне АЗ і ПЗ рівня ОС в даному випадку контролюється самим замовником. Фактично найсучаснішим способом такої інфраструктури з метою зменшення впливу описаних вище чинників є ВЗ обчислювальних систем.

Основними причинами цього є:

– різноманітність технологічних бізнес-платформ (неможливо переоцінити максимально гнучке застосування обчислювальних засобів для вирішення різного роду задач);

– відмовостійкість рішення із застосуванням технологій ВЗ;

– ймовірність, що обладнання використовується неефективно, а також наявність простоїв (коли навантаження обчислювальних засобів часто розподілене часово нерівномірно - довгі періоди низького навантаження, котрі становлять значну частину роботи всієї системи, і періоди, коли відмічається пікове зростання навантаження);

– низькощільне застосування обчислювальних процесів на одиницю АЗ;

– покращена інформаційна безпека. ВЗ дозволяє забезпечити додатковий рівень захисту від витоку інформації;

– проблема поновлення ПЗ, яка полягає в швидкості його первісного встановлення, постачання оновлень, їх застосування, а також в швидкості відновлення функціонування системи у разі збоїв.

В основі цієї робот лежить дослідження сучасних технологій ВЗ та розробка методу впровадження цих технологій в юридичній ІС з метою вирішення останньої з перерахованих вище проблем.

Стандартний підхід до встановлення клієнт-серверного додатка обмежує можливість швидкого його встановлення і налаштування, регулярного і оперативного оновлення, а також додавання нових модулів даного ПЗ, яке буде розміщено на стороні арбітражних судів.

Метою роботи є скорочення витрат часу на процеси, описані вище, за допомогою технологій ВЗ додатків і ОС.

Виходячи з проблематики і мети роботи були виділені наступні завдання:

– дослідження і порівняння існуючих на сьогодні підходів і технологій ВЗ;

– дослідження і порівняння платформ контейнерної ВЗ;

– розробка вдосконаленого підходу і технологічного процесу постачання, встановлення і запуску множини екземплярів клієнт-серверного додатка для роботи юридичної ІС і його модулів за допомогою використання технологій контейнерної ВЗ шляхом модифікації додатка і його адаптації до даного підходу.

# 1 АНАЛІЗ ДОСЛІДЖУВАНОЇ ОБЛАСТІ

З плином часу поняття ВЗ стало охоплювати досить широкий спектр технологій, хоча в кінцевому рахунку їх зміст завжди зводиться до оптимальному і гнучкішого використання наявних фізичних ресурсів, незалежно від того ВЗ це додатків (Java, Microsoft .NET Framework), носіїв інформації (RAID, SAN), ресурсів комп'ютерних мереж (NAT, VLAN) або навіть цілих комп'ютерних систем. Саме ВЗ останнього типу є об'єктом розгляду даного розділу, і нижче, якщо не вказано інше, мова буде йти саме про неї.

## 1.1 Історія концепції ВЗ

Ще в 1960-х роках 20 століття технології ВЗ були розроблені і використані компанією IBM на мейнфреймах. Власне потреба їх створення була викликана тим, що сама обчислювальна система була дорогою і рідкісною складовою, а для задоволення зростаючих потреб усіх користувачів такої системи була необхідна технологія використання спільно ресурсів для проведення обчислень. Тому з’явився механізм розподілення ресурсів обчислення між різними користувачами через застосування кожним з них ВМ (іншими словами - відокремленого логічного середовища). Таким чином віртуально кожен з користувачів був одноосібним користувачем системи, оскільки їх обчислювальні процеси були відокремленими. Отже мейнфрейм застосовувався достатньо ефективно.

Ситуація із зростанням популярності ВЗ тривала десь до 1980-х років, коли почала переважати клієнт-серверна архітектура. Зв'язано було це з широким використанням архітектури процесора x86, котра спочатку не була призначена для вирішення питання ВЗ, а також із здешевленням вартості обладнання, котре базувалося на цій архітектурі.

Проте із збільшенням продуктивності і удосконаленнями процесорів властиво питання застосування ресурсів раціонально знову постало. Відповідно

до [1], «компанія IDC, котра проводить дослідження ринку, абсолютна більшість серверів x86 задіюють в своїй роботі в середньому усього від 15 до 19 відсотків від усіх обчислювальних ресурсів». Тобто, технологія ВЗ знову почала ставати достатньо актуальною. Факти стартували сучасні розробки в цій галузі з подолання обмежень архітектури x86, яка не була призначена для ВЗ, вже потім винайшли процесори з підтримкою апаратної ВЗ для машин –клієнтів та серверів.

За способами ВЗ поділяється на дві великі групи:

– ВЗ рівня ОС;

– ВЗ на базі ВМ.

## 1.2 ВЗ на основі ВМ

Загальноприйнято, що ВМ є ізольованим ПЗ або АЗ для проведення певних обчислень та розрахунків [3]. Реалізовуватися такі ВМ можуть по-різному.

ВМ – це певний програмний контейнер, котрий може здійснювати імітацію роботи якогось окремого комп'ютера, котрий володіє своєю власною ОС і набором додатків, а також різні системні пристрої (вінчестер, процесор чи оперативна пам'ять). Такий засіб, за визначенням, є ізольованим абсолютно повністю від зовнішнього світу та інших віртуальних комп’ютерів, процес імітації роботи його пристроїв забезпечується тільки завдяки відповідному ПЗ.

Щоб бути достатньо точним, варто навести декілька зауважень до суті самої природи ВМ та властиво дійсно повної їх ізоляції від впливу зовнішніх факторів, зокрема:

– різні засоби ВЗ дають змогу ВМ одержувати прямий доступ до окремих пристроїв, котрі призначені для користувача (зокрема USB-накопичувачів, частин вінчестерів чи навіть цілих логічних дисків);

– властиво ВМ досить часто входять до складу вже існуючих комп'ютерних мереж як повноцінні члени чи навіть спеціальних віртуальних мереж, котрі містять в своєму складі тільки ВМ;

– сучасні покоління процесорів мають спеціалізовані команди та структури даних для побудови ВМ і успішного керування ними.

На сучасному етапі розвитку ІТ використовуються такі типи ВЗ на базі ВМ:

– гостьова ОС;

– паралельна ВМ;

– на базі ГВ:

– паравіртуалізація;

– повна ВЗ;

– ВЗ на рівні ядра.

Варто розглянути кожен з них у більшому об’ємі.

ВЗ гостьової ОС. В такій ВЗ будь-яка ВМ є окремим елементом ОС всередині програми, котра відповідає за ВЗ, яка управляється визначеною ОС. Яскравими прикладами ПЗ, котре реалізує цю ідею, є: Parallels Workstation [11], VMWare Workstation [12], Oracle VirtualBox [13]. Визначена ОС, під керуванням котрої працює ПЗ ВЗ, носить назву хост-системи (від host - господар), так як ця ОС дає додаткове, котре власне і забезпечує ВЗ, саме середовище виконання процесів.

Паралельна ВМ. Декілька реальних або віртуальних обчислювальних пристроїв складають при допомозі підходів кластеризації одну ВМ (як приклад, пакет PVM). Одержаний кластер зможе проводити складні обчислення та операції, котрі зв’язані із постійною роботою з дисковими даними, завдяки спільному використанню його обчислювальних складових [3].

ВЗ на основі ГВ. Цей підхід передбачає, що до складу архітектури повинна входити додаткова частина, котра носить назву ГВ [4], що є легковагним з високою оптимізацією процесів програмним прошарком між АЗ та ПЗ. Із використанням термінів концепції кілець захисту така частина встановлюється в нульовому кільці (див. рис. 1.1), хоча це, як правило, характерне для ОС. Відповідно, ОС реального обчислювача і ВМ опрацьовуються в кільцях з меншими привілегіями.

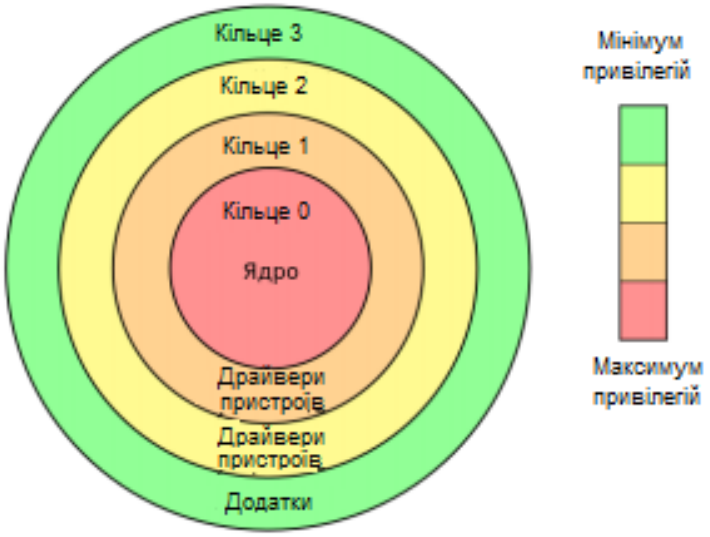


Рисунок 1.1 – Концепція кілець

ГВ призначений для виконання декількох основних задач.

По-перше, для розпізнавання, перехоплювання і надання відповідей на різні команди із визначеними привілегіями. Пізніше вони емулюються ГВ, так ніби ці командні фрагменти були опрацьовані від імені визначеної ОС. Такий спосіб носить назву зниження рівня привілеїв кільця [2].

По-друге, упорядкування, перенаправлення і повернення результатів конкретних запитів від ОС до різноманітного обладнання. Управляюча ОС виконується, власне, в середовищі ГВ - аналогічно, як і всі ВМ, ОС застосовується для керування ГВ і ВМ.

Отже ГВ властиво розширює можливість поєднаного використання обчислювальних потужностей множиною користувацьких задач і планує процесорний час як це здійснює і будь-яка ОС, але для ГВ самими споживачами ресурсів обчислення є цілі ОС, а не окремі програмні додатки. А це вже ускладнює задачу.

Перший тип ГВ не потребує хостової ОС, до його складу входять вбудовані драйвери пристроїв і планувальник. На рис. 1.2 наведена конструкція системи, яка використовує такий ГВ.

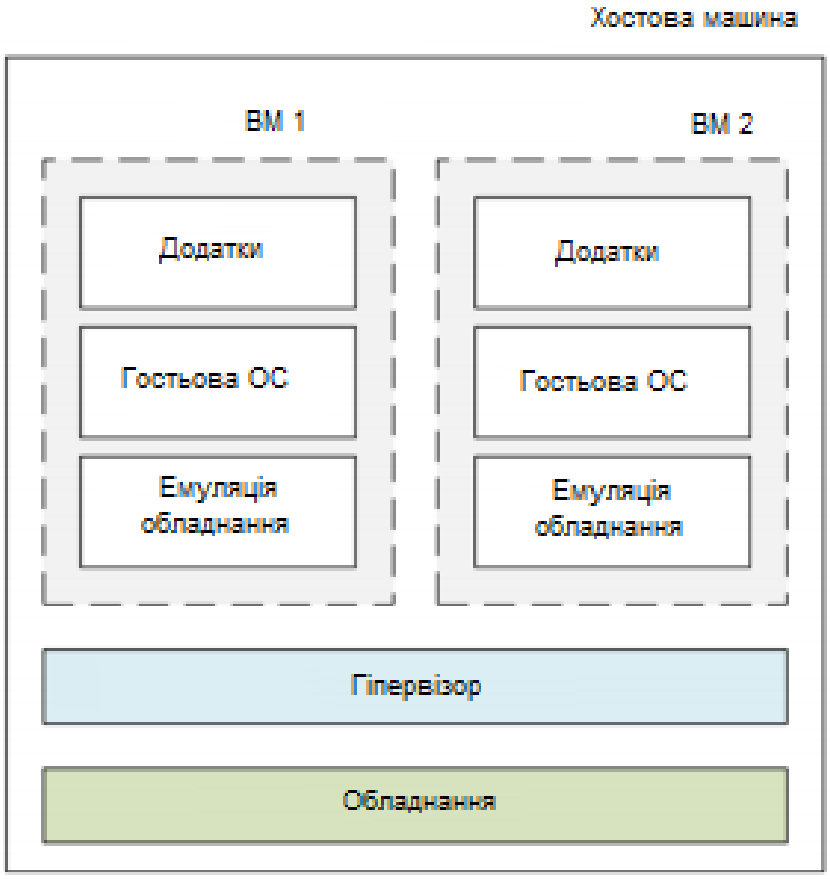


Рисунок 1.2 – Перший тип ГВ

ГВ ж другого типу виконує свою роботу зверху хостової ОС і бере для використання її функції для проведення і/о операцій, керування пам'яттю і інших [4]. На рис. 1.3 показана конструкція системи, що використовує такий ГВ.

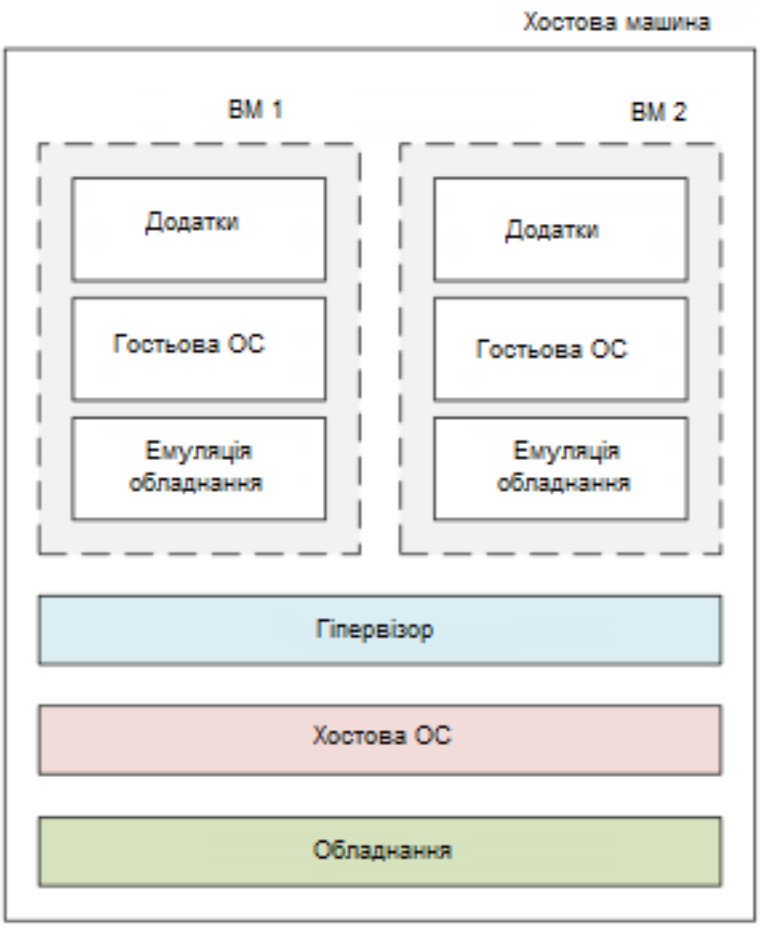


Рисунок 1.3 – Другий тип ГВ

Розглянемо вподальшому підтипи ВЗ на основі ГВ. Їх є декілька.

Паравіртуалізація. Це підхід, при використанні якого необхідне внесення поточних змін до ядра ОС, котра працюватиме в середовищі ГВ. Застосовуючи паравіртуалізацію, можна значно покращити продуктивність відносно до застосуванням інших концепцій, тому що поточні зміни, котрі виконуються власне в ядрі ОС, надають їй можливість працювати з ГВ напряму. Проте, така потреба внесення поточних змін в ядро ОС є серйозною завадою, котра не дозволяє запустити багато типів ОС на ГВ.

Основними програмними продуктами для ВЗ є Microsoft Hyper-V [15], Xen [15], VMWare ESX Server [12].

Системи на основі паравіртуалізації найбільше досліджуються в цій кваліфікаційній роботі.

Повна ВЗ. При такому підході ГВ містить програмний код, котрий забезпечує прозору для ВМ емуляцію наявного обладнання. Це, власне, і уможливлює запуск навіть немодифікованих ОС. Саме повну ВЗ використовують програмні засоби VMWare ESX Server [12] та Xen [13].

Такий підхід гіпервізорної ВЗ так само називають класичним.

### 1.3 Контейнерна ВЗ

Класична (повна) ВЗ, як було описано раніше, грунтується на використанні ГВ. Технологія контейнерної ВЗ є так званою ВЗ рівня ОС. Вона не застосовує функції ГВ, але забезпечує одночасне запускання на одному АЗ кількох екземплярів однієї ОС [3]. Такі ОС застосовують ядро хостової ОС, власне їх одночасна спільна робота стає можливою за рахунок різноманітних механізмів ізоляції та поділу ресурсів, відповідальними за які є ядро хостової ОС та так званий віртуалізаційних шар. На рис 1.4. наведено архітектуру системи, котра застосовує контейнерну ВЗ.

Використання такого підходу має безліч переваг:

– прозорість. При повній ВЗ кожна гостьова система має повний стек обладнання (віртуального), тоді як всі контейнери працюють, використовуючи безпосередньо обладнання хостової ОС;

– продуктивність. Так як при використанні контейнерів зникає необхідність трансляції системних викликів гостьових систем, емуляції роботи устаткування і багатьох інших ресурсоємних операцій, істотно зростає продуктивність гостьових систем, побудованих із використанням технології контейнерної ВЗ;

– компактність. Для роботи контейнера не потрібно повноцінного примірника ОС, а, отже, контейнер займає значно менше дискового простору у порівнянні з образом ВМ;

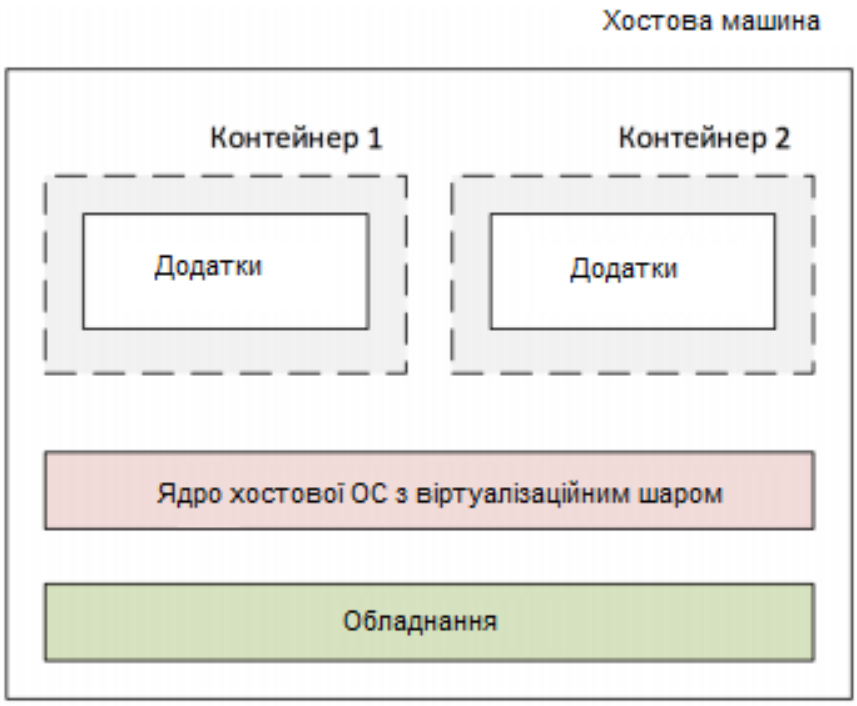


Рисунок 1.4 – Контейнерна ВЗ

– зручність управління. Для забезпечення взаємодії між контейнерами і управління їх роботою немає необхідності використовувати різні сторонні механізми.

Проте є і ряд обмежень технології контейнерної ВЗ:

– на поточний момент в основному активно розвивається тільки для ОС сімейства Linux;

– теоретично забезпечує менший рівень ізоляції, а, отже, і безпеки гостьових систем.

Продуктивність. Як відомо, використання ВЗ негативно позначається на продуктивності гостьової системи. Це падіння продуктивності обумовлено необхідністю виконання додаткових операцій, котрі забезпечують роботу ВМ. Прийнято оцінювати це падіння продуктивності у відсотках. Величина, що показує наскільки зменшується продуктивність системи при приміщенні її в віртуальне середовище називається оверхед (overhead). Як показують дослідження, проведені компаніями HP, IBM і ін., оверхед для систем контейнерної ВЗ становить 0.1-1% порівняно з показниками систем повної ВЗ.

Також, системи, поміщені в контейнери, значно виграють у віртуалізованих в таких показниках як час запуску, затримка роботи з системами введення / виведення, що робить їх застосування більш вигідним в деяких сферах, де ці показники критичні.

Таким чином, контейнерна ВЗ пропонує зовсім інший підхід до спільного використанню ресурсів хостової ОС. В той час як гостьові системи мають у використанні загальне ядро, вони повністю ізольовані одна від одної за допомогою namespaces. Це дозволяє досягти економії ресурсів, яка неможлива при використанні класичної ВЗ. При цьому, механізми ізоляції є досить захищеними, а за продуктивністю системи, розвернуті в контейнерах, не поступаються повноцінним ВМ.

# 2 ПРОЕКТНА ЧАСТИНА

## 2.1 Програмні технології для контейнерної ВЗ

### 2.1.1 OpenVZ

Це реалізація контейнерної ВЗ рівня ОС від фірми Parallels. ПЗ потребує для роботи спеціальний дистрибутив ядра Linux. Варто відмітити, що керування контейнерами проходить з хостової ОС, яку іноді називають нульовим контейнером [7].

Контейнери в OpenVZ можуть називатися також гостьовим оточенням.

Важливим є те, що кожен контейнер володіє доступом тільки до свого набору ресурсів, котрий може бути знизу і зверху обмеженим. Формування та побудова контейнерів здійснюється під дією спеціальних контейнерних шаблонів. OpenVZ надає змогу створити і провести налаштування групи поточних ресурсів, а також додатково відокремлені від них квоти (дискові, на споживання CPU) та пріоритети операцій введення / виведення. Практично всі поточні обчислювальні ресурси можуть гарантуватися і не гарантуватися (незадіяні поточні ресурси можуть перерозподілятися між контейнерами). Фактично ймовірні і значні обмеження на значну їх частку. Властиво якраз і наявність для окремого контейнера таких поточних ресурсів забезпечує сервісну стійкість, тоді як негарантовані ресурси ймовірно нададуть продуктивність, оскільки надають змогу забирати всю потужність фізичного сервера (при його простої) для сервісів. Властиво номери усіх процесів усередині контейнерів віртуалізовані: pid процесу init всередині контейнера у будь-якому випадку рівне одиниці, проте на хостовій системі цей номер є іншим, аналогічно до інших процесів поточного контейнера.

Ця технологія надає фактично повний мережевий стек для поточного контейнера. Дозволено застосовувати такі мережеві інтерфейси:

– 30 venet, котрий є продуктивнішим;
– veth, котрий підтримує Ethernet mac-адреси для контейнерів.

### 2.1.2 Linux Containers (LXC)

Технологія є дуже близькою до OpenVZ, проте при цьому існує одна важлива відмінність - все потрібно для використання LXC уже є впровадженим у стандартне Linux- ядро. Власне LXC є об'єднання використання технологій ядра Linux, котрі дозволяють забезпечити ізоляцію різноманітних ресурсів і набір утиліт для управління контейнерами.

Керування ресурсами, по суті, відбувається так само, як і в OpenVZ, проте виконується вбудованим в ядро спеціальним механізмом cgroups. Сама ізоляція наявного простору процесів, мережевого стека і інших механізмів роботи ОС здійснюється завдяки технології namespaces. Всі ці технології будуть використовуватися надалі в роботі.

Схема організації контейнерів в LXC зображена на рисунку 2.1.

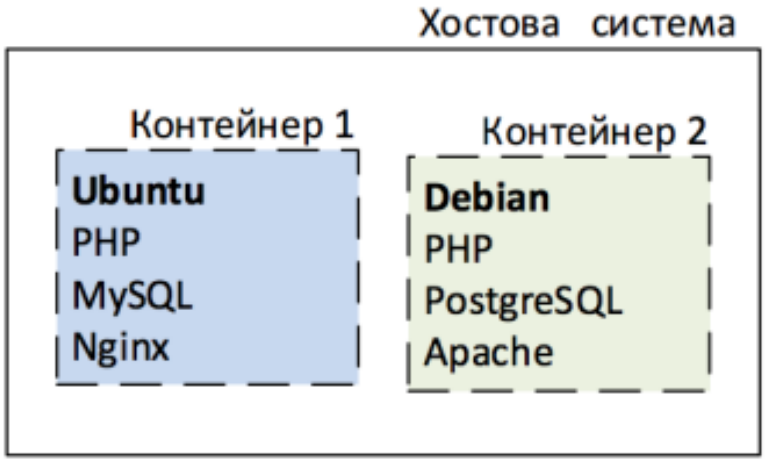


Рисунок2.1 – Контейнери LXC

На сучасному етапі розробки OpenVZ і LXC є тісно пов'язаними між собою, так як технології фактично є двома реалізаціями того самого підходу [16].

### 2.1.3 Docker

Також є системою ВЗ рівня ОС, в основі якої лежить застосування контейнерів. Першочергово основою Docker була LXC, яка була описана вище, і був зручним та достатньо гнучким інтерфейсом керування контейнерами разом

із системою контролю версій. Це, власне, і робило Docker значно зручнішим при застосуванні контейнерної ВЗ для розгортання і тестування додатків. Незважаючи на те, що ця технологія спочатку базувалася на технології LXC, в Docker дещо відмінна архітектура створення віртуальної системи: на відміну від в LXC, в якій один контейнер містить все дерево процесів, котре пов'язане з гостьовою ОС, в Docker додатки, за такої нагоди, заключаються в окремі контейнери з обмеженою взаємодією, названі шари (layers).

З однієї сторони, це надає змогу для гнучкішого настроювання самих контейнерів, проте з іншого робить суттєво складнішою архітектуру системи [9]. Архітектура Docker-контейнерів показана на рисунку 2.2.

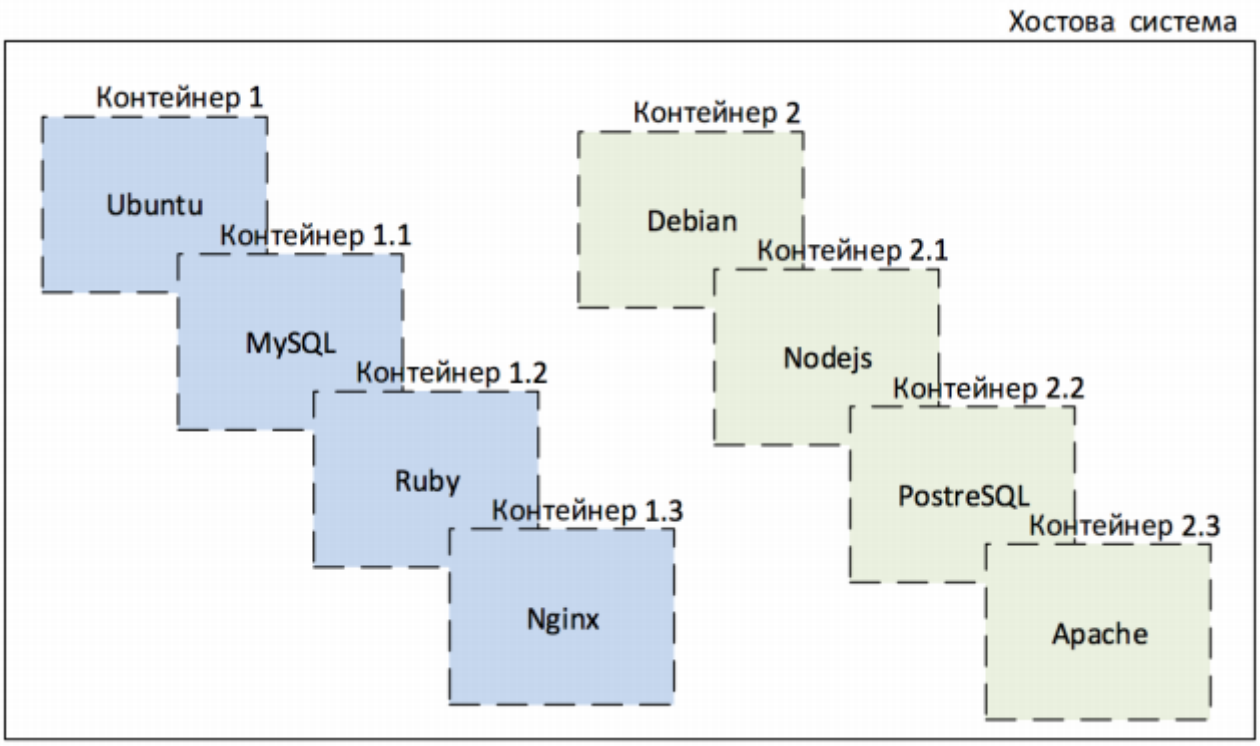


Рисунок 2.2 – Організація контейнерів Docker

Як підсумок, можна сказати, що як технологія контейнерної ВЗ була обрана саме платформа Docker. Серед розглянутих рішень саме вона найбільш задовільняє вимоги, поставлені в задачах на проектування. У порівнянні з OpenVZ, Docker є більш простою і зрозумілою, тому що використовує вбудовані

в ядро Linux механізми. Також Docker активно підтримується як розробниками, так і спільнотою користувачів.

## 2.2 Порівняння ВМ з контейнерною ВЗ

Підводячи підсумки до сказаного вище, варто виділити ключові відмінні риси ВМ і контейнерів.

Використання підходу контейнерної ВЗ дозволяє встановити набагато більше примірників додатків на один фізичний сервер, що вкрай важливо в наш час, коли все більшого значення набуває ніша хмарних обчислень, де все залежить від ресурсів дата-центрів. ВМ споживають велику кількість ресурсів. Кожна ВМ є не тільки повною копією ОС, але і повною віртуальною копією АЗ, яке потрібно щоб підтримувати працездатність цієї ОС. Це тягне за собою часте звертання до оперативної пам'яті і процесору фізичного сервера. У порівнянні, контейнеру потрібна лише ОС, системні ресурси, а також набір бібліотек, необхідних для запуску того чи іншого ПЗ. На практиці це означає, при використанні контейнерної ВЗ ми можемо розмістити на одному фізичному сервері в два або навіть в три рази більше додатків, ніж при використанні ВМ.

На додаток до наведеного вище за допомогою підходу контейнерної ВЗ стає можливим створення портативного і консистентного оточення для розробки, тестування та постачання ПЗ, що є великим плюсом при порівнянні з ВМ.

Також, ключовими факторами у виборі підходу є метрики часу, що витрачається на запуск, встановлення і відновлення системи після збоїв. У випадку контейнерів цей час зводиться до секунд, в той час як ВМ потрібні хвилини, а то й куди більший час, в разі первісного встановлення всієї системи.

Основні відмінності даних концептів ВЗ можна побачити в таблиці 2.1 [5, 6].

Таблиця 2.1 – Відмінності підходів ВЗ

| | ВМ | Контейнер |
|---|---|---|
| Рівень Вр | АЗ сервера | ОС |
| Абстракція | ОС від АЗ | Додаток від ОС |
| Гостьове оточення | Кожна гостьова система має своє ядро ОС і бібліотеки | Використовують одне ядро ОС і іноді бібліотеки |
| Час завантаження | Хвилини | Секунди |
| Продуктивність | Обмежена конфігурацією | Не залежить від конфігурації |
| Розмір образу | Велика вага, повноцінна система | Легковагове рішення |

Проаналізувавши перераховані вище критерії і порівняння приходимо до висновку, що для вирішення завдання, поставленого в якості мети даної роботи, оптимальним рішенням буде використання підходу контейнерної ВЗ. Таким чином, вся подальша робота буде проводитися на основі концепту контейнерної ВЗ з використанням технології Docker, так як технологія Docker являється найбільш досконалою і орієнтованою на роботу з додатками, серед всіх розглянутих раніше платформ.

Далі мова піде безпосередньо про платформу Docker, щоб отримати глибші знання про цю технологію.

## 2.3 Технологія Docker

### 2.3.1 Опис технології

Згідно з документацією, Docker є відкритою технологією для розробки, постачання та запуску програмних додатків. Вона надає можливість відокремити самі додатки від інфраструктури, що дозволяє постачати ці програми швидше в

порівнянні з традиційним підходом без використання Docker. Використовуючи переваги методологій Docker по постачанню, тестуванню та встановленню програмних продуктів, можна забезпечити значне зменшення часу між етапом розробки продукту і його запуском [9].

Docker надає можливість пакувати і запускати додатки в ізольованих оточеннях званих контейнерами, за допомогою технології контейнерної ВЗ. Ізольованість і захищеність контейнерів дозволяє запускати їх велику кількість одночасно на одній хостовій системі. Завдяки легковагості контейнерів, які запускаються без додаткового навантаження на ГВ, на одній конфігурації обладнання надається можливість одночасно запустити більшу кількість контейнерів, ніж могло б бути запущено ВМ.

Docker пропонує інструменти та платформу для керування життєвим циклом контейнерів:

– інкапсуляція додатків і їх компонентів в Docker контейнери;

– можливість створювати дистрибутиви контейнерів для їх подальшої передачі і зберігання;

– можливість розгортати контейнери додатків на різній інфраструктурі, будь це локальний дата-центр або хмара.

В основі Docker лежить архітектура, відома як «клієнт-сервер» (див. рис. 2.3). Тобто, сам користувач не має змоги взаємодіяти напряму з сервером, а має скористатися Docker-клієнтом з цією метою. Такий клієнт опрацьовує отримані від користувача спеціальні команди та програмно співпрацює з Docker-демоном з використанням командного рядка або RESTful API. Даний демон перебирає на себе функції побудови Docker об'єктів і функції керування ними. До об'єктів Docker можуть відноситься образи контейнерів, самі контейнери, опису топології мереж, накопичувачі даних.

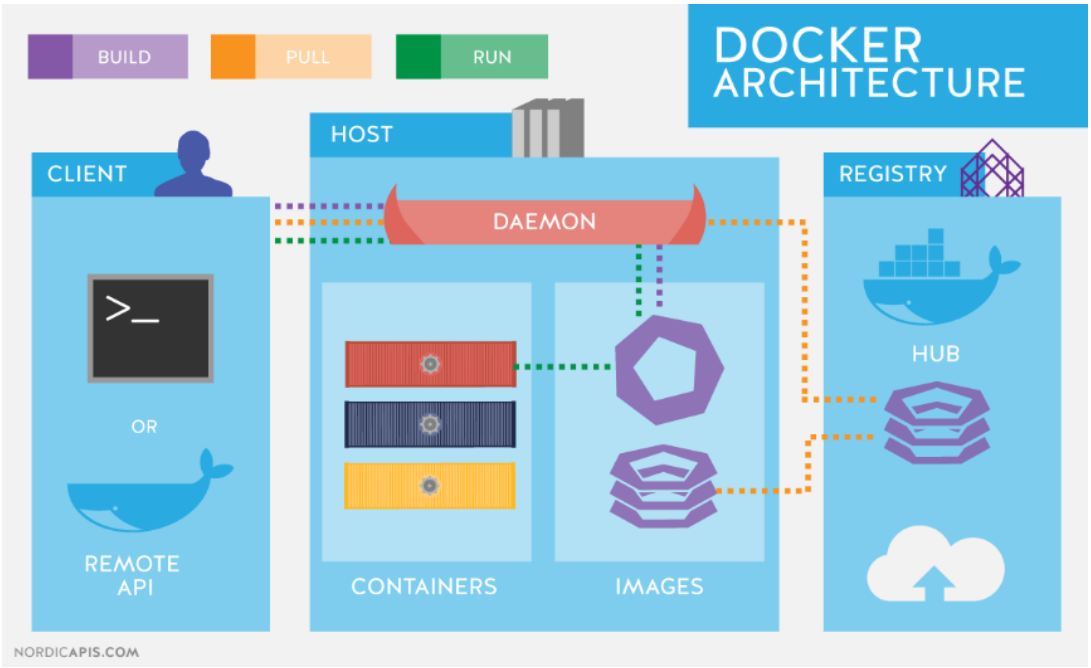


Рисунок 2.3 – Архітектура Docker

Діаграма звернень зображена на рисунку 2.4.

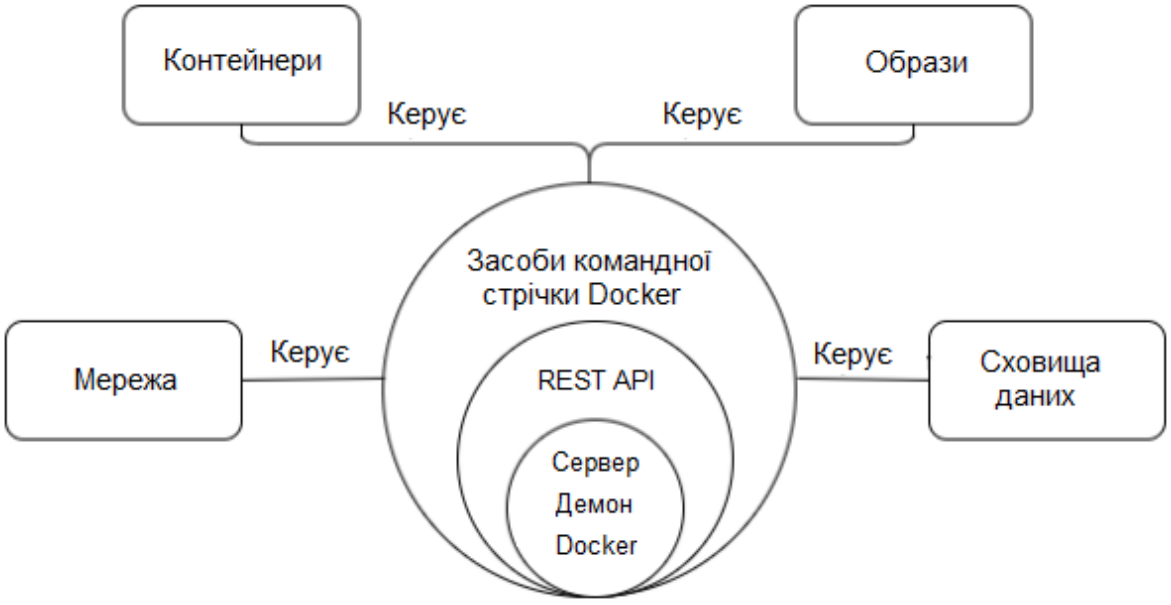


Рисунок 2.4 – Звернення в Docker

### 2.3.2 Образи Docker

Є спеціальними шаблонами з атрибутом «тільки читання» з інструкціями побудови Docker контейнерів [9]. Для прикладу, образ може мати ОС Ubuntu

Linux з встановленим в ній веб-сервером Apache. Дозволено образи збирати і оновлювати як з нуля, так і використовуючи завантажені образи, які були створені іншими розробниками. Образ може бути заснований або успадковуватися від одного або декількох інших образів. Опис способу міститься в текстовому файлі з назвою Dockerfile, який має простий синтаксис.

Варто відмітити, що окремий образ містить певну кількість шарів. Щоб зібрати дані шари в єдиний образ Docker користається технологію UFS, котра надає змогу файлам і папкам з різних розділених ФС прозоро зв'язуватися один з одним, будуючи, таким чином, когерентну ФС. Застосування цих шарів є однією з переваг легковагості Docker.

При зміні способу, наприклад, при оновленні програми, створюється новий шар і замінює собою тільки той шар, який повинен був бути оновлений. В такому випадку користувачам не потрібно заново оновлювати весь образ в своєму робочому оточенні, досить лише отримати описане вище оновлення. Фундаментом для певного образу є визначений батьківський або базовий образ. Для прикладу, ubuntu є батьківським образом Ubuntu, чи fedora є батьківським образом Fedora. Аналогічно образи можна застосовувати як базу при побудові свіжих образів. Як варіант, можна скористатися образом apache як батьківським для роботи з веб-додатками.

Визначений набір кроків опису побудови образів носить назву інструкції. Властиво кожна інструкція призначена для побудови нового образу чи рівня.

Інструкціями є наступні дії:

- запуск команди;
- додавання файлу або директорії;
- створення змінної оточення;
- вказівки, що запускати, коли запускається контейнер цього образу.

Всі інструкції знаходяться у файлі Dockerfile. Будь-який Docker отримує інформацію з нього, коли користувач збирає образ, забезпечує виконання записаних в ньому інструкцій, а потім вертає кінцевий образ, створений за допомогою інструкцій. В рисунку 2.5 показано приклад простого Dockerfile.

```
FROM docker/whalesay:latest
RUN apt-get -y update && apt-get install -y fortunes
CMD /usr/games/fortune -a | cowsay
```

Рисунок 2.5 - Dockerfile

В даному файлі описаний образ, який успадковується від останньої доступної версії образу з назвою docker / whalesay і розширює його з допомогою додавання команд установки додаткового ПЗ і його запуску.

### 2.4 Технології, які застосовує Docker

Мова Go є основою для написання Docker. Кожен Docker бере всі наявні для застосування можливості ядра Linux для реалізації своїх функцій [9].

Docker застосовує namespaces, щоб забезпечити ізольоване робоче середовище контейнера. Завдяки namespaces для кожного контейнера може будуватися окреме дерево процесів з окремим простором імен. Кожен раз при запуску контейнера програмно створюється набір namespaces для нього.

Список використаних просторів імен:

– pid: для ізоляції процесів;

– net: для управління мережевими інтерфейсами;

– ipc: для управління IPC ресурсами. (ICP: InterProccess Communication);

– mnt: для управління точками монтування;

– utc: для ізолювання ядра системи і контролю версій (UTC: Unix timesharing system).

Docker також залежить від іншої технології, названої cgroups або контрольні групи. Дана технологія гарантує механізм, при якому додатку надаються тільки необхідні для його коректної роботи ресурси.

cgroups мають змогу розділяти наявні ресурси технічного забезпечення і, за необхідності, застосовувати різні межі і обмеження для додатку. Для прикладу, обмежити ймовірний контейнерний об’єм необхідної пам'яті.

ФС UnionFS працює, будуючи різні рівні та роблячи її дуже швидкою та легковаговою. Docker застосовує UnionFS для побудови блоків, з яких власне контейнер і створюється. Можуть застосовуватися різні варіанти UnionFS, зокрема: AUFS, btrfs, vfs і DeviceMapper.

Як короткий підсумок, Docker об’єднує різноманітні технології, котрі були перераховані вище (зокрема namespaces, cgroups, UnionFS) в одній обгортці, званій форматом контейнера. Стандартним форматом для контейнера є libcontainer. Аналогічно Docker має змогу застосовувати стандартний контейнерний формат в Linux.

### 2.5 Інсталяція Docker і Docker-репозиторія

Як було описано вище, технологія Docker потребує інсталяції серверного ПЗ для своєї роботи. Також присутня можливість для інсталяції локального репозиторія для зберігання образів Docker, що дозволяє обмежити доступ до них і не викладати їх в глобальну мережу. Далі буде розглянуто процес інсталяції і налаштування як технології Docker, так і сховища Docker образів.

#### 2.5.1 Інсталяція Docker

В даний момент технологія Docker доступна в двох варіаціях: Community Edition («Громадська версія») і Enterprise Edition («Промислова версія»). Вподальшому буде розглянуто процес інсталяції версії Community Edition, так як можливостей даного рішення досить для виконання поставленого завдання.

Технологія Docker в версії Community Edition може бути встановлена на серверному обладнанні, яке використовує такі ОС:

- Ubuntu;
- Debian;

– CentOS;
– Fedora;
– Microsoft Windows 10;
– macOS;

Далі буде розглянута інсталяція платформи Docker при використанні ОС Ubuntu версії 14.04 (LTS).

Перед встановленням платформи Docker слід переконатися, що в системі немає раніше встановлених файлів цієї системи. Даний крок необхідно виконати для усунення можливого конфлікту версій і помилок інсталяції при наступних кроках налаштування і встановлення платформи Docker. На цьому кроці досить запустити команду видалення модулів Docker в командному рядку, яка автоматично видалить компоненти Docker, якщо вони будуть знайдені в системі. Якщо дані компоненти вже відсутні в системі, то команда успішно завершитися з повідомлень що модулі Docker були знайдені. Сама команда і результат її виконання представлені на рисунку 2.6.

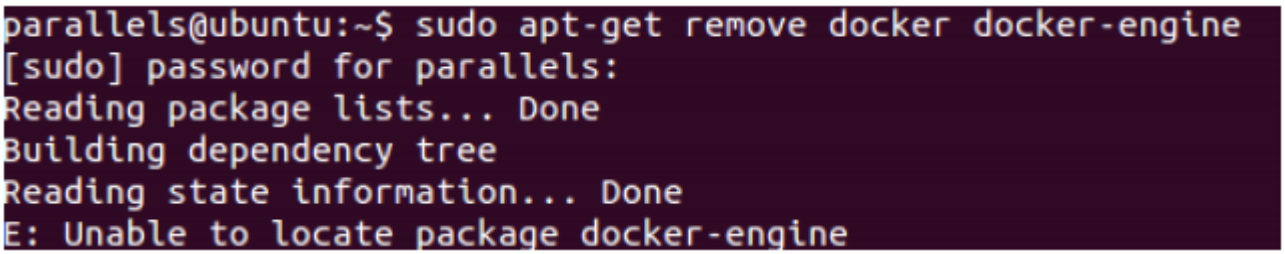


Рисунок 2.6 - Видалення пакетів Docker

Для безпосередньої інсталяції, розробниками платформи Docker рекомендується використовувати спеціальний репозиторій настановних пакетів linux. Даний репозиторій необхідно додати в список використовуваних репозиторіїв менеджера настановних пакетів ОС для його подальшого використання. Також, перед використанням даного сховища з метою інсталяції платформи Docker потрібно встановити додаткові модулі менеджера настановних пакетів ОС, щоб мати можливість використовувати протокол HTTPS для обміну інформацією з репозиторієм, і спеціальний ключ для

розшифровування отриманих даних. Команди для встановлення необхідних пакетів і ключа наведені на рис. 2.7 і 2.8 відповідно.

```
parallels@ubuntu:~$ sudo apt-get install     apt-transport-https     ca-certific
ates     curl     software-properties-common
```

Рисунок 2.7 - Встановлення https пакетів

```
parallels@ubuntu:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | su
do apt-key add -
```

Рисунок 2.8 - Встановлення GPGключа

Для безпосереднього додавання сховища в список використовуваних менеджером пакетів репозиторіїв необхідно виконати команду, представлену на рис. 2.9.

```
parallels@ubuntu:~$ sudo add-apt-repository \
>    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
>    $(lsb_release -cs) \
>    stable"
```

Рисунок 2.9 - Додавання сховища

Після оновлення списку репозиторіїв необхідно запустити команду, показану на рис. 2.10, що зміни вступили в силу і менеджер пакетів міг успішно почати інсталяцію.

```
root@ubuntu:/home/parallels# sudo apt-get update
```

Рисунок 2.10 - Оновлення індексів

Для запуску інсталяції платформи Docker необхідно запустити команду, котра відображена на рисунку 2.11.

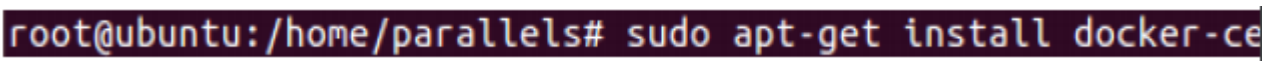


Рисунок 2.11 - Інсталяція Docker

Після завершення виконання останньої команди платформа Docker буде встановлена в системі. Офіційна документація Docker пропонує активувати тестовий контейнер з тестовими образом для перевірки того, що інсталяція була виконана успішно і платформа Docker працює коректно.

Даний тестовий контейнер можна запустити за допомогою команди, представленої на рис. 2.12.

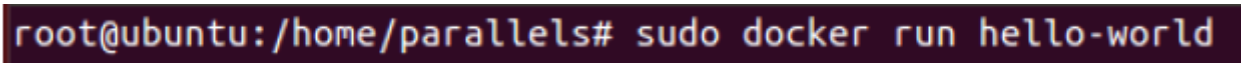


Рисунок 2.12 - Перевірка роботи Docker

Після успішного виконання наведеної вище команди запуску тестового контейнера платформа автоматично виконає завантаження образу контейнера з сайту Docker і запустатить його. При цьому в консоль виводитися інформаційне повідомлення, після чого контейнер завершує свою роботу.

Повний процес інсталяції платформи Docker при використанні ОС Ubuntu на серверному обладнанні був описаний вище. Як проміжний крок налаштування платформи, варто відзначити можливість її автоматичного запуску при старті системи. Автоматичний запуск платформи і її контейнерів може бути особливо корисним у разі непередбачених перезавантажень сервера, на якому ця платформа розгорнута.

Для включення можливості автоматичного запуску достатньо запустити в командному рядку команду, представлену на рис. 2.13.


Рисунок 2.13 - Включення автоматичного запуску Docker

### 2.5.2 Реєстр Docker

Є масштабованим серверним додатком, який має змогу зберігати образи Docker контейнерів для їх подальшого використання на серверах. Реєстр Docker може бути використаний для таких цілей:

– повний контроль на тим, де зберігаються образи Docker;

– повний контроль над самими образами і правами доступу до них;

– інтеграція системи зберігання образів Docker в процес розробки без використання сторонніх сервісів

Так як реєстр Docker є серверним додатком, його так само можна запустити в Docker контейнері. Необхідно відмітити, що сам образ реєстру вільно поширюється через глобальний, загальнодоступний реєстр Docker, котрий має назву Docker Hub.

Для запуску реєстру потрібно виконати наступну команду в системі, де вже проінстальована платформа ВЗ Docker (рисунок 2.14):

```
root@ubuntu:/home/parallels# docker run -d -p 5000:5000 --name registry registry:2
```

Рисунок 2.14 - Запуск реєстру Docker

Після успішного виконання цієї команди реєстр буде запущений і доступний при застосуванні порту 5000.

Docker Hub є хмарним сервісом, котрий призначений для організації спільної роботи, забезпечення автоматизації процесу виконання різного роду завдань, а також для створення, коректного поширення та запуску на виконання додатків, котрі адаптовані для Docker. Фактично, Docker Hub володіє набором служб, зокрема:

– забезпечення поширення контейнерного образу;

– керування поточними змінами у проекті;

– забезпечення співпраці між користувачами ПЗ та його розробниками;

– постійний супровід життєвого циклу;

– кооперація з зовнішніми службами.

Фінансова сторона використання сервісу Docker Hub така ж сама, як і у GitHub, тобто опрацювання загальнодоступних проектів не потребує оплати, тільки за необхідності застосування закритих репозиторіїв необхідно платити.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ДОДАТКА НА ПРИКЛАДІ МОДУЛЯ ВІДОБРАЖЕННЯ ФАКТІВ

## 3.1 Опис модуля відображення фактів

Модуль відображення фактів юридичної справи є частиною юридичної інформаційно системи і призначений для відображення ключових слів в тексті судових актів, що дозволяє користувачам (в даному випадку суддям) в короткий термін отримати всю ключову інформацію судового акту. Також даний модуль може бути використаний як повністю незалежний додаток і надавати свою функціональність незалежно від усієї юридичної інформаційної системи. Далі в ході роботи цей модуль буде називатися додатком.

Додаток відображення фактів спроектовано та програмно реалізовано із застосуванням клієнт-серверної архітектури. Роль сервера виконує ПЗ розроблене за допомогою технології Spring Boot.

Дана технологія дозволяє запускати серверне ПЗ на будь-якому серверному обладнання, де встановлено середовище виконання Java версії 7 і вище. Подібна можливість досягається завдяки вбудованому контейнеру додатків Tomcat, який постачається в одному пакеті з самим серверним додатком і не вимагає встановлення.

Роль клієнта виконує веб-додаток, розроблений за допомогою технології Angular версії 4.0. Даний клієнт є односторінковим веб-додатком, всі ресурси якого завантажуються в браузер користувача при першому зверненні до цього додатку. Незважаючи на те, що додаток є односторінковим, він не є статичним, а надає користувачеві різну функціональність, зокрема:

- перегляд списку всіх актів;
- вибір акту для перегляду;
- перегляд тексту акту;
- перегляд списку виділених фактів з їх категоріями;
- можливість виділення обраного факту в тексті судового акту.

Вся перерахована вище функціональність досягається за допомогою змін DOM моделі веб-сторінок і звернень до серверного ПЗ при різних діях користувача.

Клієнтське і серверне ПЗ обмінюються інформацією за допомогою протоколу HTTP і способу формату обміну даними JSON.

Нижче наведені знімки екранів додатка відображення фактів (рис. 31, 3.2).

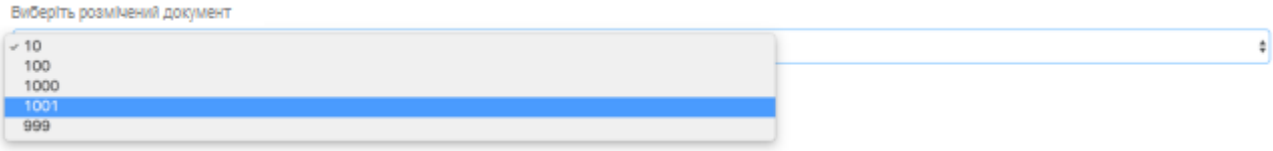


Рисунок 3.1 – Екран вибору акту для перегляду

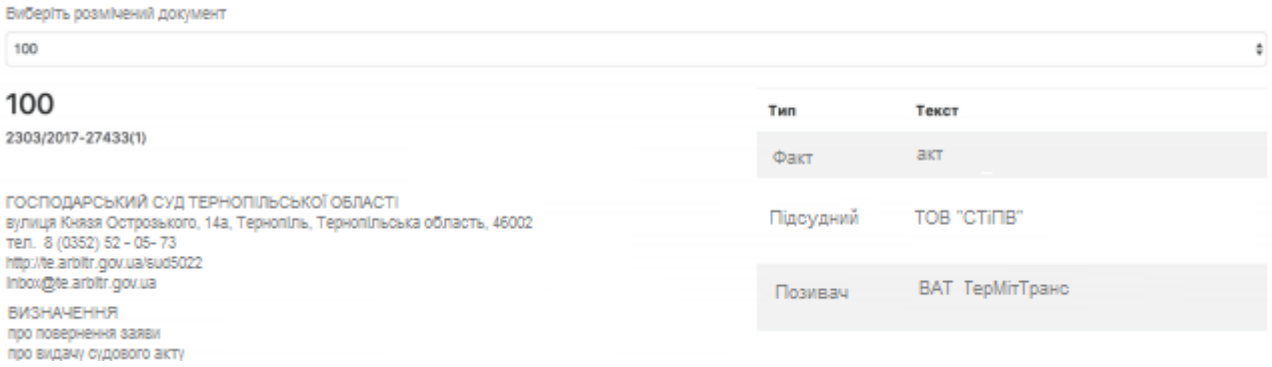


Рисунок 3.2 – Екран перегляду акта і фактів

### 3.2 Створення образу контейнера

У цьому розділі буде розглянуто підготовку так званих шарів або рівнів ВЗ, які були описані в розділі, присвяченому платформі контейнерної ВЗ Docker, і створення кінцевого образу для контейнера модуля відображення фактів юридичної інформаційної системи з використанням цих шарів.

Для створення необхідного образу достатньо трьох рівнів ВЗ.

Першим рівнем ВР виступає ОС. Для збільшення швидкості запуску контейнера і зменшення розміру його образу використовується дистрибутив ОС

linux, котрий носить назву Alpine. Для порівняння, образ контейнера Alpine зі встановленим в системі mysql клієнтом займає 16 мегабайт дискового простору, в той час як ідентичний йому образ ОС Ubuntu з встановленим mysql клієнтом займає 232 мегабайта. Назвемо даний рівень зі встановленою на ньому ОС - environment-base. В лістингу 3.1 показано Dockerfile даного рівня образу.

Лістинг 3.1 - Dockerfile 1 шару ВЗ «environment-base»

```
FROM scratch
MAINTAINER Stepan Ozh
LABEL name = "environment-base" \
description = "Base image with Alpine OS for remote code
compilation and execution on docker
container project "
ADD rootfs.tar.gz /
```

У представленому Dockerfile описані інструкції додавання структури ФС в docker-контейнер. При створенні контейнера ОС ініціалізується з файлової структури, яка міститься в підготовленому архіві. Рішення використовувати заздалегідь підготовлений архів у форматі * .tar.gz обумовлено відсутністю необхідності вводити залежність від стороннього батьківського Dockerfile, який надає Alpine в DockerHub (відкритий реєстр Docker). Також з'являється можливість, при необхідності, змінити пакет ФС, не змінюючи структури шарів для існуючих додатків.

Другим рівнем, який також можна назвати другим шаром docker-контейнера, є середовище для роботи з вихідним кодом на мові програмування Java, необхідне для запуску серверного ПЗ. Назва даного шару - environment-jdk-7u80. На цьому рівні необхідно розгорнути пакет розробки JavaDevelopmentKit, який включає в себе інструменти компіляції і виконання вихідного Java -коду. В лістингу 3.2 наведено фрагмент Dockerfile для даного шару контейнера.

Лістинг 3.2 - Фрагмент Dockerfile 2 шару ВЗ environment-jdk07u80

```
FROM localhost: 5000 / base: latest
MAINTAINER Stepan Ozh
LABEL name = "environment-jdk-7u80" \
description = "Image with JDK installation on top of Alpine
OS"
# Java Version and other ENV
ENV JAVA_VERSION_MAJOR = 7 \
JAVA_VERSION_MINOR = 80 \
JAVA_VERSION_BUILD = 15 \
JAVA_PACKAGE = jdk \
JAVA_JCE = standard \
GLIBC_VERSION = 2.23-r3 \
LANG = C.UTF-8
# Do all in one step
RUN set -ex && \
apk upgrade --update && \
apk add --update libstdc ++ curl ca-certificates bash && \
for pkg in glibc - $ {GLIBC_VERSION} glibc-bin - $
{GLIBC_VERSION} glibc-i18n - $ {GLIBC_VERSION};
do curl -sSL https://github.com/andyshinn/alpine-pkg-
glibc/releases/download/${GLIBC_VERSION}/${pkg}.apk -o
/tmp/${pkg}.apk; done && \
apk add --allow-untrusted /tmp/*.apk && \
rm -v /tmp/*.apk && \
…
```

Образ другого рівня успадковується від образу першого рівня, який включається в себе ОС, як було описано вище, таким чином, всі дії здійснюються при ініціалізації другого шару docker-контейнера можуть бути виконані з допомогою вбудованих засобів доступною ОС. В даному Dockerfile описується процес створення базової директорії для установки JDK, завантаження і розпакування офіційного інсталяційного пакета JDK (Java Development Kit) з сайту Oracle за допомогою спеціально встановленої для цього утиліти curl, видалення установчого пакета JDK і утиліти curl після успішної ініціалізації засобів розробника Java і встановлення змінних оточення JAVA_HOME і PATH,

Третім шаром docker-контейнера модуля відображення фактів є рівень серверного ПЗ, який забезпечує користувачу заявлену в попередньому розділі функціональність. На даному рівні необхідно лише сформувати рядки з командою перенесення копії додатки в docker-контейнер та командою запуску додатка, яка буде виконана при ініціалізації контейнера. Так як даний шар

успадковується від шару з ім'ям -environment-jdk-7u80, в ньому вже містяться встановлена ОС Linux Alpine і середовище виконання вихідного коду Java. Цей рівень ВЗ отримав назву «lowyer-fact-module-app». Лістинг 3.3 відображає даний Dockerfile.

Лістинг 3.3 - Dockerfile 3 шару ВЗ «lowyer -fact-module-app»

```
FROM localhost: 5000 / jdk
MAINTAINER Stepan Ozh
LABEL name = "jurist-fact-module-app" \
description = "Image with fact module installation"
VOLUME / tmp
ADD facts-module-demo.jar app.jar
RUN sh -c 'touch /app.jar'
ENV JAVA_OPTS = ""
ENTRYPOINT [ "sh", "-c", "java $ JAVA_OPTS -jar /app.jar"]
```

Трьох перерахованих вище шарів docker-контейнера досить для запуску додатка відображення фактів в судових актах. Далі розглянемо кроки збірки, встановлення і запуску цього docker-контейнера з використанням підготовлених Dockerfile для кожного шару.

Для побудови та встановлення в реєстр всіх шарів docker-контейнера потрібно забезпечити виконання наступних кроків у командному рядку:

– перейти в директорію з Dockerfile першого шару і виконати команду: dockerbuild -tlocalhost: 5000 / base;

– перейти в директорію з Dockerfile другого шару і виконати наступну команду: dockerbuild -tlocalhost: 5000 / jdk;

– перейти в директорію з Dockerfile третього шару (останнього для даного контейнера) і виконати таку команду: dockerbuild -tlocalhost: 5000 / facts-app.

Після збірки і інсталяції в реестр Docker останнього образу може бути запущений docker-контейнер для цього образу. Для запуску контейнера з будь-якої директорії в командному рядку треба виконати команду:

dockerrun --namefacts-app-p 8080: 8080 localhost: 5000 / facts-app .

У даній команді з допомогою аргументу –name передається ім'я контейнера, яке буде використовуватися для зручності звернення до нього при виконанні різних команд, передбачених платформою Docker. При допомозі параметра –p 8080: 8080 встановлюється порт, який буде доступний образ Docker-контейнера.

Після виконання всіх перерахованих вище команд docker-контейнер модуля відображення фактів в судових справах буде запущений і готовий до використання. Для того, щоб скористатися додатком, запущеному в docker-контейнері досить в браузері хостової ОС перейти за адресою http: // localhost: 8080, або замінити localhost на IP-адресу хостової ОС, щоб мати можливість отримати доступ до додатку з іншого комп'ютера в мережі.

### 3.3 Налаштування системи автоматичного складання і публікації образів

У попередньому підрозділі було розглянуто процес створення Dockerfile інструкцій і образів контейнерів за допомогою засобів командного рядка Docker. Варто відзначити, що перші два з описаних вище рівнів контейнеризації не пов'язані з самим додатком, який буде запущено в docker-контейнері, і відповідно, образи цих контейнером будуть змінюватися вкрай рідко. Третій же рівень контейнеризації представляє собою сам додаток, який в процесі розробки може досить часто змінюватися, що вимагає повторного створення образу з командного рядка при кожній зміні програми та завантаження нового образу в реєстр Docker.

Для автоматизації цього процесу було вирішено скористатися наявними можливостями засобу зборки і управління залежностями Maven, котрий використовується в розробці модуля відображення фактів інформаційної юридичної системи.

Технології Maven дозволяє скомпонувати новий образ контейнера і завантажити його в реєстр Docker на етапі складання всього проекту. При такому

підході розробник може оновити образ контейнера не відходячи від свого звичного порядку дій при складанні модуля, що позитивно впливає на швидкість розробки. Засіб Maven використовує конфігураційний файл XML- формату, для управління залежностями і процесом зборки проекту, який називається pom.xml. Для того щоб використовувати можливість автоматичного збираня і публікації образів необхідно додати налаштування, зображені на рисунку 3.3, в секцію build конфігурації pom.xml.

```
<build>
    <plugins>
        <plugin>
            <groupId>com.spotify</groupId>
            <artifactId>docker-maven-plugin</artifactId>
            <version>0.4.11</version>
            <configuration>
                <imageName>${project.artifactId}</imageName>
                <dockerDirectory>src/main/docker</dockerDirectory>
                <resources>
                    <resource>
                        <targetPath>/</targetPath>
                        <directory>${project.build.directory}</directory>
                        <include>${project.build.finalName}.jar</include>
                    </resource>
                </resources>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Рисунок 3.3 – Додаткові налаштування в pom.xml

Після застосування нової конфігурації для збірки модуля відображення фактів, створення нового образу контейнера і завантаження цього контейнера в реєстр Docker достатньо запустити наступну команду: mvnpackagedocker: build -DpushImage.

### 3.4 Порівняння тимчасових метрик після застосування технології Docker

У цьому розділі буде виконано порівняння часу, який витрачається на інсталяцію і оновлення модуля відображення фактів юридичної інформаційної системи при використанні підходу контейнерної ВЗ Docker і використанні ВЗ за допомогою ВМ.

Для початку порівняємо час, що витрачається на повне встановлення модуля. Діаграма розподілу часу наведена на рис. 3.4. Інсталяція нового модуля за допомогою технології Docker-контейнерів зайняла в цілому 15 хвилин, в той час як інсталяція того ж модуля на серверне ПЗ з ВЗ за допомогою ВМ зайняла 45 хвилин. Це пояснюється тим, що ВМ вимагає більше часу на встановлення віртуальної ОС і ПЗ, яке потрібно для коректного функціонування модуля.

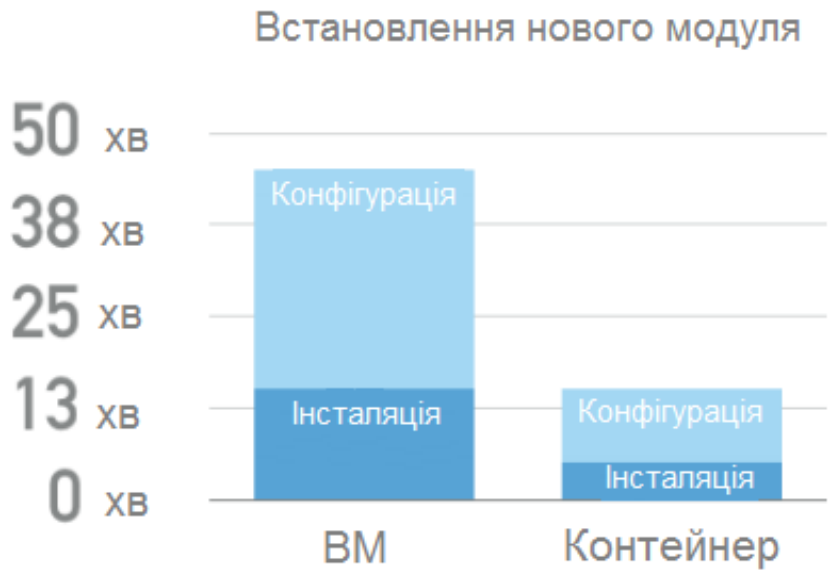


Рисунок 3.4 – Витрачений час на встановлення

Порівняння витрат часу на оновлення модуля відображення фактів юридичної інформаційної системи зображено на рисунку 3.5.

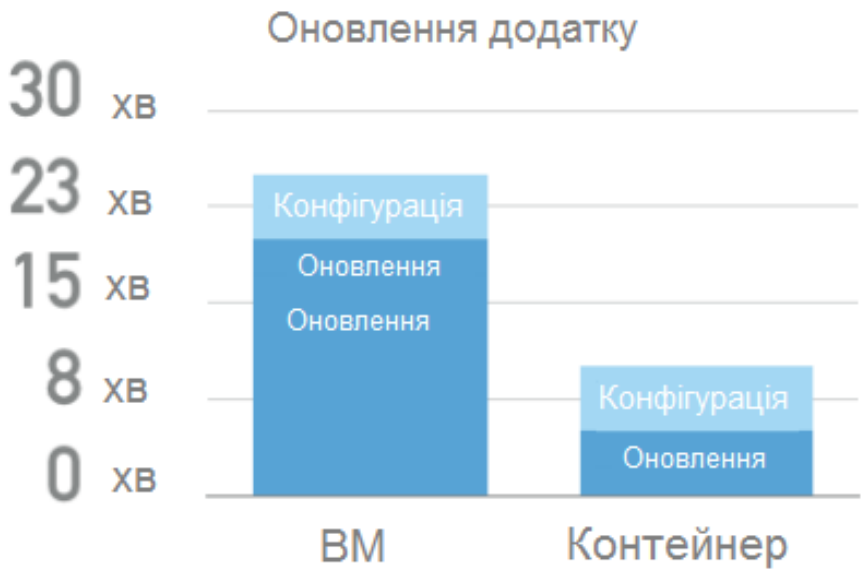


Рисунок 3.5 – Витрачений час на оновлення

На ВМ оновлення модуля займає 20 хвилин, а у випадку зміни будь-яких налаштувань або конфігураційних файлів час поновлення зростає до 25 хвилин, на сервері, де використовується технологія контейнерної ВЗ Docker ті ж дії займають відповідно 5 і 10 хвилин. Така різниця пояснюється тим, що при оновленні модуля на ВМ необхідно під’єднюватися безпосередньо до самої машини і вручну виконувати кожен крок оновлення ПЗ, працюючи з віртуальною ОС та її ОС. Оновлення модуля, що знаходиться в Docker-контейнері, може бути виконано за допомогою будь-якого комп'ютера в мережі, на якому встановлені засоби командного рядка Docker.

Велика частина часу в такому випадку витрачатиметься на мережеві активності передачі інформації, які не вимагають втручання розробників і користувачів. Як видно з діаграми на рисунку 3.5, конфігурація оновленого модуля може зайняти приблизно однакову кількість часу, так як в разі ВМ може виникнути необхідність працювати файлами на самій ВМ з встановленим модулів, а у випадку з віртуальною контейнеризацією Docker може знадобитися зміни інструкційDockerfile, що втім є вкрай рідкісним сценарієм.

Таким чином, використовуючи платформу контейнерної ВЗ Docker, можна отримати трикратне зменшення витрат часу на такі базові операції над ПЗ як його початкове встановлення і подальші оновлення.

# 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

## 4.1 Санітарно-гігієнічні вимоги до умов праціз ПК

Санітарні правила і норми влаштування і обладнання кабінетів комп'ютерної техніки в навчальних закладах та режиму праці учнів на персональних комп'ютерах встановлюють нормативи фізичних чинників, що створюються комп'ютерами при їх роботі, та гігієнічні вимоги до проектування, виготовлення і експлуатації вітчизняних та експлуатації імпортних персональних комп'ютерів, що застосовуються в навчально-виховному процесі.

Вимоги до приміщень та розташування робочих місць з ПК: приміщення, призначені для роботи з ПК, повинні мати природне освітлення. Орієнтація вікон повинна бути на північ або північний схід, вікна повинні мати жалюзі, які можна регулювати, або штори; не дозволяється розміщувати кабінети обчислювальної техніки у підвальних приміщеннях будинків; кабінети, обладнані комп'ютерною технікою, в навчальних закладах повинні розміщуватись в окремих приміщеннях з природним освітленням та організованим обміном повітря; стіни, стеля і підлога та обладнання кабінетів комп'ютерної техніки повинні мати покриття із матеріалів з матовою фактурою з коефіцієнтом відбиття: стін — 40- 50 %, стелі — 70 - 80 %, підлоги — 20-30 %, предметів обладнання — 40-60 % (робочого столу — 40-50 %, корпуса дисплею та клавіатури — 30-50 %, стелажів — 40-60 %); поверхня підлоги повинна мати антистатичне покриття та бути зручною для вологого прибирання; забороняється використовувати для оздоблення інтер'єру приміщень комп'ютерних кабінетів полімерні матеріали (дерев'яно-стружкові плити, шпалери, що придатні для миття, плівкові та рулонні синтетичні матеріали, шаровий паперовий пластик та ін.), що виділяють у повітря шкідливі хімічні речовини, які перевищують гранично допустимі концентрації; вміст шкідливих хімічних речовин в повітрі дошкільних та учбових приміщень з комп'ютерною технікою не повинен перевищувати середньодобові концентрації [17].

Вимоги до освітлення приміщень та робочих місць: приміщення з ПК повинні мати природне та штучне освітлення; штучне освітлення в приміщеннях з ПК повинно здійснюватись системою загального освітлення; як джерела світла при такому освітленні повинні застосовуватись переважно люмінесцентні лампи; яскравість світильників загального освітлення в зоні кутів випромінювання від 50° до 90° з вертикаллю в поздовжній та поперечній площинах повинна складати не більше 200 кд/м$^2$, захисний кут світильників повинен бути не менше 40; загальне освітлення повинно бути виконано у вигляді суцільних або переривчастих ліній світильників; штучне освітлення повинно забезпечувати на робочих місцях в кабінетах з ПК освітленість не нижчу, а на екранах дисплеїв — не вище приведених в таблиці 4.1; коефіцієнт запасу для освітлювальних установок загального освітлення приймається рівним 1,4; необхідно проводити чищення скла вікон та світильників не менше двох разів на рік, а також заміну перегорілих ламп по мірі їх виходу з ладу; в кабінетах з ПК слід обмежити нерівномірність розподілу яскравості в полі зору учнів [18]. Співвідношення яскравості між робочим екраном та близьким оточенням не повинно перевищувати 5:1, між поверхнями робочого екрану і оточенням (стіл, обладнання) — 10:1; величина коефіцієнту пульсації освітленості не повинна перевищувати 5 %. Газорозрядні лампи повинні застосовуватись в світильниках загального та місцевого освітлення з високочастотними пускорегулюючими апаратами; необхідно передбачити обмеження прямого блиску від джерел природного та штучного освітлення; яскравість великих поверхонь (вікна, світильники і таке інше), що знаходяться у полі зору, не повинна перевищувати 200 кд/м$^2$, мірою захисту від прямого блиску має бути зниження яскравості видимої частини джерел світла застосуванням спеціальних розсіювачів, відбивачів та інших світлозахисних пристроїв, а також правильне розміщення робочих місць відносно джерел світла; повинні передбачатись заходи щодо обмеження відбитого блиску на робочих поверхнях (екран, стіл, клавіатура); яскравість полисків на екрані не повинні перевищувати 80 кд/кв. м. Яскравість

стелі при застосуванні системи відбитого освітлення не повинна перевищувати 200 кд/кв. м.

Таблиця 4.1 — Норми освітленості в кабінетах з ПК

| Характеристика роботи | Робоча поверхня | Площина | Освітленість,лк | Примітка |
|---|---|---|---|---|
| Робота переважно з екранами дисплеїв ПК (50 % та більше робочого часу) | Екран | вертикальна | 200 | не вище |
| | Клавіатура | горизонтальна | 400 | не нижче |
| | Стіл | горизонтальна | 400 | не нижче |
| Робота переважно з екранами дисплеїв ПК (менше 50 % робочого часу) | Екран | вертикальна | 200 | не вище |
| | Клавіатура | горизонтальна | 400 | не нижче |
| | Стіл | горизонтальна | 500 | не нижче |
| | Дошка | вертикальна | 500 | не нижче |
| Проходи основні | Підлога | горизонтальна | 100 | |

Вимоги, що забезпечують захист від впливу іонізуючих та неіонізуючих електромагнітних полів та випромінювань: ВДТ на електронно-променевих трубках можуть бути потенційними джерелами гігієнічно значимих рівнів електромагнітних випромінювань в діапазоні частот 50Гц-300 МГц; інтенсивність ультрафіолетового випромінювання на відстані 0,3м від екрану не повинна перевищувати в діапазоні довжин хвиль 400 - 320 нм — 2 Вт/м$^2$, 320 - 280 нм — 0,002 Вт/м$^2$, ультрафіолетового випромінювання в діапазоні 280 - 200 нм — не повинно бути.

### 4.2 Вимоги до виробничого освітлення та його нормування

Приміщення для роботи з ВДТ повинні мати природне та штучне освітлення відповідно до ДБН В.2.5-28-2006 (на заміну СНиП II-4-79).

Природне освітлення має здійснюватись через світлові прорізи, орієнтовані переважно на північ чи північний схід і забезпечувати коефіцієнт природної освітленості не нижче ніж 1,5%. Розраховується він за методикою, викладеною в ДБН В.2.5-28-2006. За виробничої потреби дозволяється експлуатувати ЕОМ у приміщеннях без природного освітлення за узгодженням з органами державного нагляду за охороною праці та органами і установами санітарно-епідеміологічної служби. Вікна приміщень з ВДТ повинні мати регулювальні пристрої для відкривання, а також жалюзі, штори, зовнішні козирки та ін.

Штучне освітлення приміщення з робочими місцями, обладнаними ВДТ ЕОМ загального та персонального користування, має бути обладнане системою загального рівномірного освітлення. У виробничих та адміністративно-громадських приміщеннях, де переважають роботи з документами, допускається вживати систему комбінованого освітлення (додатково до загального освітлення встановлюються світильники місцевого освітлення).

Загальне освітлення має бути виконане у вигляді суцільних або переривчатих ліній світильників, що розміщуються збоку від робочих місць (переважно зліва) паралельно лінії зору працівників. Допускається застосовувати світильники таких класів світлорозподілу: світильники прямого світла – П; переважно прямого світла – Н; переважно відбитого світла – В. При розташуванні ВДТ за периметром приміщення лінії світильників штучного освітлення повинні розміщуватися локально над робочими місцями. Для загального освітлення необхідно застосовувати світильники із розсіювачами та дзеркальними екранними сітками або віддзеркалювачами, укомплектовані високочастотними пускорегулювальними апаратами (ВЧ ПРА). Застосування світильників без розсіювачів та екранних сіток забороняється [18].

Як джерело світла при штучному освітленні повинні застосовуватися, як правило, люмінесцентні лампи типу ЛБ. При обладнанні відбивного освітлення у виробничих та адміністративно-громадських приміщеннях можуть

застосовуватися металогалогенні лампи потужністю до 250 Вт. Допускається у світильниках місцевого освітлення застосовувати лампи розжарювання.

Яскравість світильників загального освітлення в зоні кутів випромінювання від 50º до 90º відносно вертикалі в подовжній і поперечній площинах повинна складати не більше 200 кд/м2, а захисний кут світильників повинен бути не більшим за 40º. Коефіцієнт запасу відповідно до ДБН В.2.5-28-2006 для освітлювальної установки загального освітлення слід приймати рівним 1,4. Коефіцієнт пульсації повинен не перевищувати 5 % і забезпечуватися застосуванням газорозрядних ламп у світильниках загального і місцевого освітлення. За відсутності світильників з ВЧ ПРА лампи багатолампових світильників або розташовані поруч світильники загального освітлення необхідно підключати до різних фаз трифазної мережі.

Рівень освітленості на робочому столі в зоні розташування документів має бути в межах 300…500 лк. У разі неможливості забезпечити даний рівень освітленості системою загального освітлення допускається застосування світильників місцевого освітлення, але при цьому не повинно бути відблисків на поверхні екрану та збільшення освітленості екрану більше ніж до 300 лк. Світильники місцевого освітлення повинні мати напівпрозорий відбивач світла з захисним кутом не меншим за 40º. Необхідно передбачити обмеження прямої блискості від джерела природного та штучного освітлення, при цьому яскравість поверхонь, що світяться (вікна, джерела штучного світла) і перебувають у полі зору, повинна бути не більшою за 200 кд/м2. Необхідно обмежувати відбиту блискість шляхом правильного вибору типів світильників та розміщенням робочих місць відносно джерел природного та штучного освітлення. При цьому яскравість відблисків на екрані відеотерміналу не повинна перевищувати 40 кд/м2, яскравість стелі при застосуванні системи відбивного освітлення не повинна перевищувати 200 кд/м2 [17].

Необхідно обмежувати нерівномірність розподілу яскравості в полі зору осіб, що працюють з відеотерміналом, при цьому відношення значень яскравості

робочих поверхонь не повинно перевищувати 3:1, а робочих поверхонь і навколишніх предметів (стіни, обладнання) – 5:1.

Необхідно використовувати систему вимикачів, що дозволяє регулювати інтенсивність штучного освітлення залежно від інтенсивності природного, а також дозволяє освітлювати тільки потрібні для роботи зони приміщення.

Для забезпечення нормованих значень освітлення в приміщеннях з відеотерміналами ЕОМ загального та персонального користування необхідно очищати віконне скло та світильники не рідше ніж 2 рази на рік, та своєчасно проводити заміну ламп, що перегоріли.

# ВИСНОВКИ

При виконанні кваліфікаційної роботи були розглянуті методи апаратної та контейнерної ВЗ, а саме:

– було проведено дослідження та порівняння підходів і технологій ВЗ;

– було проведено дослідження і порівняння технологій контейнерної ВЗ.

В ході дослідження підходів ВЗ були розглянуті такі технології, як контейнерна ВЗ і ВЗ, заснована на використанні ВМ. В результаті порівняння цих підходів було прийнято рішення використовувати технологію контейнерної ВЗ для вирішення проблеми, зазначеної в темі в даній роботі.

Глибшому розгляду були піддані платформи контейнерної ВЗ, такі як OpenVZ, LXC і Docker. В результаті дослідження платформ ВЗ і порівняльного їх аналізу, для виконання поставленого завдання була обрана технологія контейнерної ВЗ Docker, так як вона є найбільш оптимальної для вирішення заявленої проблеми.

В результаті роботи був розроблений удосконалений підхід і технологічний процес поставки, встановлення і запуску множини примірників клієнт-серверного додатка юридичної фірми і його модулів з допомогою використання платформи контейнерної ВР Docker, що дозволило значно скоротити часові витрати на процеси інсталяції і оновлення модулів інформаційної юридичної системи, а саме:

– тимчасові витрати на оновлення модулів зменшилися в 2 рази;

– тимчасові витрати на установку модулів зменшилися в 3 рази.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. History of virtualization, virtual machines, server consolidation [Електронний ресурс] // VMWare, Inc., Palo Alto, California, USA - Режим доступу: http://www.vmware.com - (дата звертання: 05.05.2021).

2. Рівні привілеїв гіпервізора [Електронний ресурс] - Режим доступу: http://upload.wikimedia.org/wikipedia/en/thumb/2/2f/Priv_rings.svg/633pxPriv_ rings.svg.png - (дата звертання: 10.05.2021).

3. PC Magazine/RE: Введение в виртуализацию. Обзорный курс [Електронний ресурс]. - Режим доступу: http://www.pcmag.ru/elearning/course/ index.php?COURSE_ID=14 - (дата звертання: 09.05.2021).

4. Гипервизоры, виртуализация и облако [Електроний ресурс]. -Режим доступу: http://www.ibm.com/developerworks/ru/library/cl-hypervisorcompare, - (дата звертання: 02.05.2021).

5. Performance Evaluation of Virtualization Technologies for Server Consolidation [Електронний ресурс]. - Режим доступу: http://www.hpl.hp.com/ techreports/2007/HPL-2007-59R1.pdf - (дата звертання: 12.05.2021).

6. An Updated Performance Comparison of Virtual Machines and Linux Containers [Електронний ресурс]. - Режим доступу: http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195 DD819C852 57D2300681E7B/$File/rc25482.pdf - (дата звертання: 02.05.2021)..

7. OpenVZ Virtuozzo Containers Wiki [Електронний ресурс]. - Режим доступу: https://openvz.org/Main_Page (дата звертання: 04.05.2021).

8. Containers & Docker: How Secure Are They [Електронний ресурс]. - Режим доступу: https://blog.docker.com/2013/08/containers-docker-how-secure-arethey – (дата звертання: 04.05.2021).

9. Docker official doumentation [Електронний ресурс]. - Режим доступу: https://docs.docker.com/engine/understanding-docker - (дата звертання: 12.05.2021).

10. Spring Boot with Docker [Електронний ресурс]. - Режим доступу: https://spring.io/guides/gs/spring-boot-docker – (дата звертання: 14.05.2021).

11. Parallels Official Website [Електронний ресурс]. - Режим доступу: http://www.parallels.com/ – (дата звертання: 14.05.2021).

12. VMWare Official Website [Електронний ресурс]. - Режим доступу: https://www.vmware.com/ – (дата звертання: 02.05.2021).

13. Oracle VM VirtualBox [Електронний ресурс]. - Режим доступу: https://www.virtualbox.org/ – (дата звертання: 27.05.2021).

14. TheXenProject [Електронний ресурс]. - Режим доступу: https://www.xenproject.org/ – (дата звертання: 12.05.2021).

15. Hyper-Voverview – TechNet–Microsoft [Електронний ресурс]. - Режим доступу: https://technet.microsoft.com/en-us/library/hh831531(v=ws.11).aspx – (дата звертання: 13.05.2021).

16. LinuxContainers [Електронний ресурс]. - Режим доступу: https://linuxcontainers.org/ru/ – (дата звертання: 23.05.2021).

17. Яремко З. М. Безпека життєдіяльності: Навч. посіб. — Львів., 2005. – 301 с.

18. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. [Електронний ресурс] - Режим доступу: https://zakon.rada.gov.ua/laws/show/z0508-18 – (дата звертання: 01.06.2021).