**Теоретичне завдання**

**1. У чому принципова різниця у між контейнерами та віртуальними машинами при розміщенні проектів?**

Основна принципова різниця між ними полягає в тому, як вони ізолюють та управляють ресурсами.

Віртуальні машини емулюють окремі фізичні комп'ютери в межах одного фізичного сервера, кожна віртуальна машина має свою власну операційну систему, яка працює безпосередньо на апаратному забезпеченні сервера, використовує свої ресурси, включаючи пам'ять, процесорний час, мережеві інтерфейси та дисковий простір. Ресурси фізичного сервера розділяються між віртуальними машинами.

Контейнери також забезпечують ізольоване середовище для запуску програмного забезпечення, але вони працюють на рівні операційної системи, а не емулюють повністю окремі віртуальні машини. Контейнери використовують спільну операційну систему хоста, але мають власні ізольовані простори процесів, файлові системи та мережеві інтерфейси, ділять ядро операційної системи хоста, що дозволяє їм бути легкими та швидкими у виконанні.

Основна перевага контейнерів полягає у їхній портативності, ефективності використання ресурсів та швидкості розгортання. Контейнери дозволяють запускати додатки в уніфікованому середовищі, що спрощує розгортання і розвиток програмного забезпечення.

З іншого боку, віртуальні машини надають більшу ізоляцію та гнучкість, оскільки кожна машина може мати власну операційну систему та незалежні ресурси.

**2. З яких складових складається контейнер?**

Container Image − пакет, який містить всі необхідні компоненти для запуску додатку або сервісу. Образ включає операційну систему, виконувані файли, бібліотеки, залежності та конфігураційні файли.

Container Registry − централізоване сховище, де зберігаються контейнерні образи.

Container Orchestrator − інструмент або платформа, яка дозволяє керувати та управляти контейнерами, включаючи їх розгортання, шкалювання, моніторинг, управління мережею та балансування навантаження.

Container Runtime − програмне забезпечення, яке дозволяє запускати та управляти контейнерами.

Конфігураційні файли − використовуються для налаштування контейнера. Вони визначають параметри контейнера, такі як базовий образ, порти, залежності, змінні середовища та інші налаштування.

**3. Що таке Docker compose?**

Docker Compose − інструмент для оркестрації та управління багатоконтейнерними додатками. Він дозволяє описати структуру та конфігурацію вашого додатку у вигляді файлу YAML, який називається docker-compose.yml. Цей файл містить опис сервісів, мереж та об'єктів обсягу, необхідних для додатку.

За допомогою Docker Compose можна визначити кілька контейнерів, які взаємодіють між собою, і встановити їх конфігурацію, таку як налаштування мережі, змінні середовища, обсяги, порти тощо. Docker Compose забезпечує простий спосіб запускати та зупиняти багатоконтейнерні додатки зі збереженням їх конфігурації та параметрів.

Docker Compose є потужним інструментом для розробки, тестування та розгортання контейнерних додатків, особливо якщо потрібно керувати багатьма контейнерами, які взаємодіють між собою.

**4. Які є типи мережі при налаштуванні контейнеру?**

Основні типи мереж, які підтримуються в Docker, включають:

Default − кожен контейнер має свою власну ізольовану мережу. Вони можуть взаємодіяти лише за допомогою прив'язаних до контейнерів портів або спеціально налаштованих мережевих з’єднань.

Bridge − тип мережі, який використовується за замовчуванням при створенні контейнера. Всі контейнери, що використовують мережу типу «bridge», можуть спілкуватися між собою за допомогою імені контейнера або IP-адреси.

Host − контейнер використовує ту ж саму мережу, що й хостова система. Це означає, що контейнер має доступ до всіх мережевих інтерфейсів та портів хостової системи без потреби у прив'язці портів.

None − контейнер не має мережі, ізольований від будь-якого мережевого з'єднання. В цьому випадку контейнер може використовувати лише внутрішню комунікацію або мережеві з'єднання, які були явно налаштовані.

Custom − власна мережа з довільними налаштуваннями, створюється використовуючи Docker network CLI або Docker Compose.