Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

Лабораторна робота №8

з дисципліни «Операційна система UNIX»

«Установка Docker»

Виконав:

студент 2-го курсу, НН ІАТЕ

групи ТР-23

Ровний Григорій Олександрович

Перевірила:

проф. Левченко Л.О.

КИЇВ 2023

**Мета роботи:**

набути навичок встановлювати додаток Docker в ОС Linux, надати основи для роботи на постійній основі з образами та контейнерами, що дозволяє не засмічувати робочу машину локально встановленими різними версіями низки програмного забезпечення: apache, mysql, virtualenv, python, mongodb, memchaced, redis, php, а також подібного програмного забезпечення, яке використовується при розробці проектів та часто ще й конфліктує між собою від версії до версії.

**Теоретичні відомості**

Docker є найпопулярнішою платформою управління контейнерами. Це ПЗ з відкритим кодом, принцип роботи якого найпростіше порівняти з транспортними контейнерами. Філософію Docker часто описують за допомогою метафори «доставки універсальних вантажних контейнерів», які можна переміщувати між різними видами транспорту з мінімумом зусиль.

Така ідея була перенесена на ІТ-сферу для переміщення коду між різними програмними середовищами з мінімальними обсягами роботи. Коли розробляється додаток, необхідно надати код разом з усіма його складовими, такими як бібліотеки, сервер, бази даних і т. д. Може мати місце така ситуація, коли додаток працює на вашому комп'ютері, але відмовляється працювати на комп’ютері іншого користувача. Ця проблема вирішується через створення програмного забезпечення, яке не залежить від системи.

Термінологія

Контейнери - це технологія упаковки і запуску додатків Windows, Linux, MacOS в різних локальних середовищах і в хмарі.

Контейнер - це виконуваний екземпляр, який інкапсулює необхідну ПЗ. Він складається з образів. Його можна легко видалити і знову створити за короткий проміжок часу. Контейнери надають невимогливе до ресурсів ізольоване середовище, яке спрощує розробку, розгортання, запуск ПЗ, особливо в динамічних і розподілених середовищах та керування додатками.

Образ - базовий елемент кожного контейнера. Залежно від способу, може знадобитися деякий час для його створення.

Порт - це порт TCP/UDP (протоколи транспортного рівня для передачі пакетів між комп’ютерами) в своєму первинному значенні. Щоб все було просто, припустимо, що порти можуть бути відкриті в зовнішньому світі або підключені до контейнерів (доступні тільки з цих контейнерів і невидимі для зовнішнього світу).

Том - описується як загальна папка. Тома ініціалізуються при створенні контейнера і призначені для збереження даних, незалежно від життєвого циклу контейнера.

Реєстр - це сервер, на якому зберігаються образи. Порівняємо його з GitHub: ви можете витягнути образ з реєстру, щоб розгорнути його локально, і так само локально можете вносити в реєстр створені образи.

Docker Hub - публічний репозиторій з інтерфейсом, що надається Docker Inc. Він зберігає безліч образів. Ресурс є джерелом «офіційних» образів, зроблених командою Докер або створених у співпраці з розробником ПЗ. Ця інформація відкрита для будь-якого зареєстрованого користувача.

Контейнери створюються на основі ядра операційної системи сервера, але не отримують необмежений доступ до ядра. Наприклад, контейнер може звертатися до віртуалізованої версії файлової системи і реєстру, але будь-які зміни стосуються тільки контейнера і видаляються при його зупинці. Контейнер збирається поверх ядра, але ядро не надає всі інтерфейси API і служби, необхідні для запуску програми. Більшість з них надаються системними файлами (бібліотеками), які працюють на рівні вище ядра в режимі користувача. Оскільки контейнер ізольований від середовища режиму користувача сервера, контейнеру потрібно власна копія цих системних файлів режиму користувача, які упаковуються в базовий образ. Базовий образ виступає в якості основного рівня, на якому збирається контейнер, надаючи йому служби операційної системи, які не надаються ядром.

Таким чином, Docker використовує засоби ядра, які дозволяють створювати ізольовані групи процесів. При запуску Docker робить лише кілька системних викликів і ядро створює для нового процесу окремий простір PID-ів, окрему віртуальну мережу, окремий набір обмежень по ресурсах. Ядро асоціює Docker зі специфічним набором налаштувань.

Всі контейнери створюються з образів контейнерів. Образи контейнерів являють собою набір файлів, організованих в стек шарів, розташованих на локальному комп'ютері або у віддаленому реєстрі контейнерів. Образ контейнера складається з файлів ОС режиму користувача, необхідних для підтримки додатку, будь-яких середовищ виконання або залежностей додатків, а також будь-якого іншого файлу конфігурації, необхідного для правильної роботи додатка.

Одним з найбільш поширених варіантів використання контейнерів є мікросервіси (microservices). Мікросервіси – це спосіб розробки та компонування програмних систем, при якому вони формуються з невеликих незалежних компонентів, що взаємодіють один з одним через мережу. 64-бітовий Linux- контейнер працює тільки на хості з встановленою 64-бітовою версією ОС Linux.

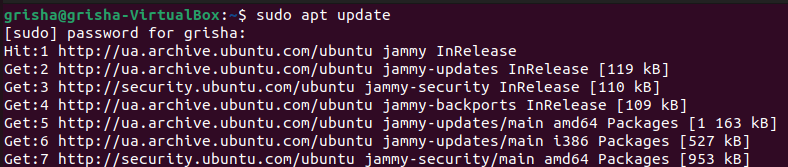
Платформа Docker складається з двох окремих компонентів: Docker Engine, механізму, що відповідає за створення і функціонування контейнерів та Docker Hub - хмарного сервісу для поширення контейнерів. Трохи пізніше були розроблені інструментальні засоби для Docker: Swarm (менеджер кластерів), Kinematic (графічний інтерфейс для роботи з контейнерами), Machine (утиліта командного рядка для підтримки роботи Docker-хостів).

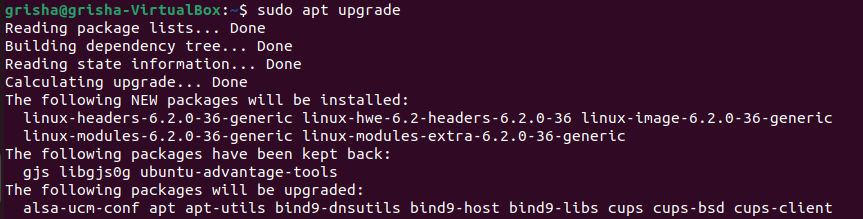
**Поставлене завдання:**

1. Встановити дистрибутив Docker.

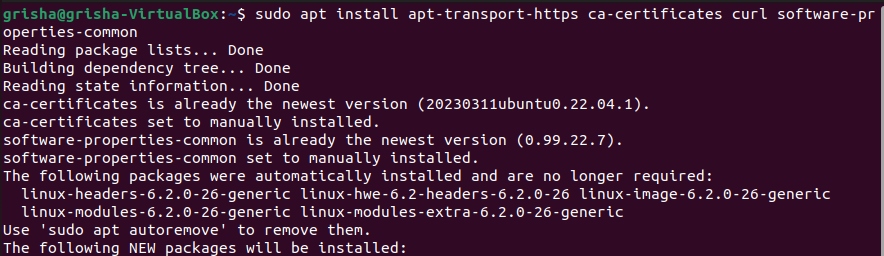
**Результат виконання роботи**

1. Крок 1: відкриваємо термінал та проводимо оновлення існуючих пакетів. Для цього скористаємося командою *sudo apt update* та *sudo apt upgrate*. Це важливо для забезпечення актуальності всіх компонентів та коректної установки програми.





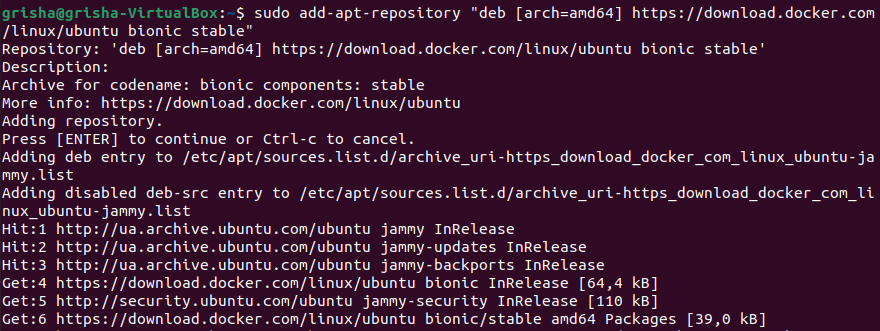
1. Крок 2: виконуємо установку пакетів, які дозволяють apt використовувати репозиторії HTTPS. Використання HTTPS для репозиторіїв є стандартом безпеки, і багато джерел програмного забезпечення вимагають використання захищеного з'єднання для забезпечення рівня безпеки. Встановити пакети можна використавши команду *sudo apt install apt-transport-https ca-certificates curl software-properties-common.*



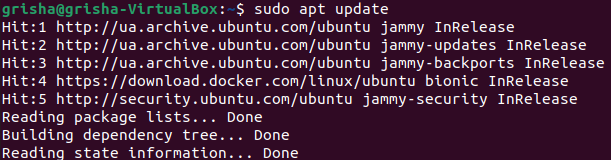
1. Крок 3: додаємо ключ GPG для офіційних репозиторіїв Docker. Ключ GPG використовується в системі для перевірки цілісності та автентичності пакету. Для установки використаємо *curl -fsSL https://download.docker.com/linux/ubuntu/gpg sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg.*



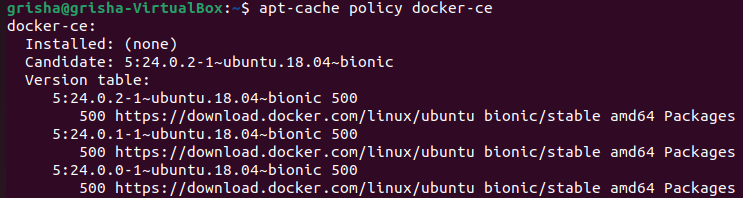
1. Далі додаємо репозиторій Docker до списку джерел APT, для того щоб система отримала інформацію де знаходяться пакети Docker та як їх отримувати: *sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"*



1. Після проведених операцій оновлюємо базу пакетів (*sudo apt update*) для додавання до них репозиторію Docker.



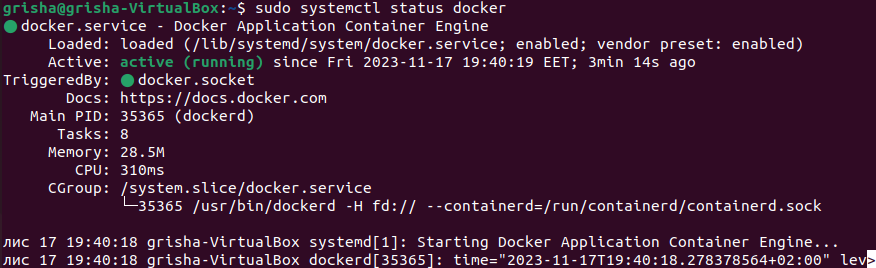
1. Слід додатково переконатися, що ми встановлюємо Docker саме з репозиторію Docker, а не з репозиторію за замовчуванням (Ubuntu). Введемо команду apt-cache policy docker-ce, після чого бачимо, що посилання вказують на сайт docker, а значить все налаштовано вірно.



1. Виконуємо установку Docker -  *sudo apt install docker-ce*.



1. Виконаємо перевірку чи процес запущений за допомогою команди *sudo systemctl status docker.* Як бачимо, ми отримали інформацію про стан служби, ID процесу, журнал служби, активний час роботи, скільки займає процес пам’яті і т.д.

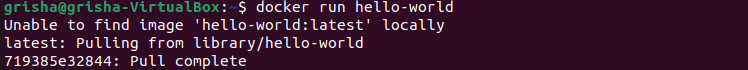


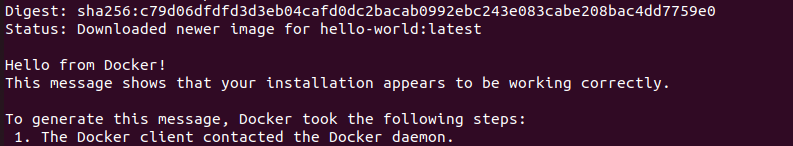
1. За замовчуванням Docker може запускати користувач тільки з правами root або з групи docker, тому зазвичай прийнято використовувати sudo. Але можливо також додати користувача у docker групу. Додамо користувача grisha у групу docker, для цього пропишемо *sudo usermod –aG docker grisha* і застосуємо ці зміни *su – grisha.*



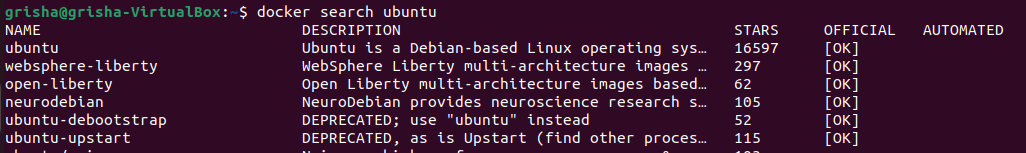


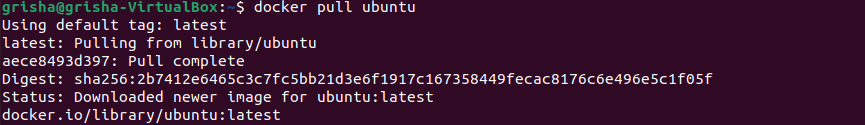
1. Запустимо тестовий образ hello-world (docker run hello-world)

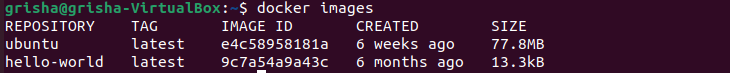




1. Спробуємо попрацювати з образами в Docker Hub. Знайдемо образ Ubuntu (docker search ubuntu) та завантажимо його (docker pull ubuntu). Спочатку ми отримаємо список всіх образів, схожих за назвою, з їх короткою характеристикою. Серед цього списку знаходимо необхідний образ та завантажуємо його. Після чого виконаємо *docker images*, щоб переконатися в тому що необхідний образ був успішно завантажений.







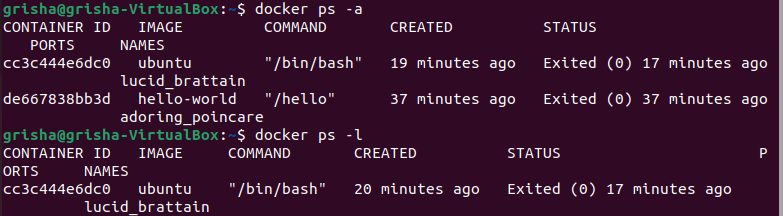
1. Виконаємо запуск контейнера на основі Ubuntu (*docker run –it ubuntu)*. Бачимо, що командний рядок змінює свій вигляд, що вказує на те, що надалі ми працюємо в контейнері. Щоб вийти з контейнера необхідно ввести команду *exit*.



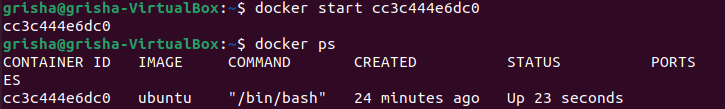


1. Переглянемо список активних контейнерів (docker ps), список всіх контейнерів (docker ps -a), та останній конейнер(docker ps -l):

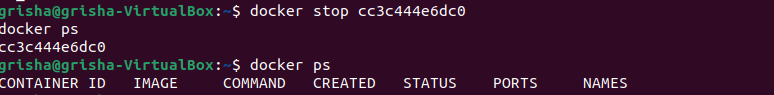




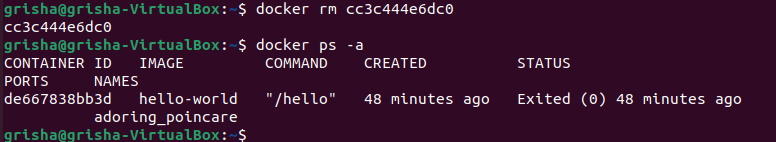
1. Запустимо останній контейнер Ubuntu, для цього в команду *docker start* передаємо його ідентифікатор ID або ім’я NAMES. Бачимо, що контейнер з’явився у списку активних.



1. Для зупинки контейнерів використовують *docker stop <ID/NAMES>.*



1. Якщо контейнер більше не потрібний, то можемо його видалити : *docker rm <ID/NAMES>.*

**

**Висновок:**

У результаті виконання лабораторної роботи було встановлено та налаштовано дистрибутив Docker; отримані основні навички для роботи з контейнерами Docker.