Міністерство освіти і науки України

НТУУ «КПІ ім. Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

Лабораторна робота №8 з дисципліни «Операційна система UNIX»

**Установка Docker**

**Варіант: 16**

Виконав студент групи ТР-12

Каркушевський Владислав

Перевірила д.т.н., проф. Левченко Л. О.

КИЇВ 2022

**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.

**Теоретична частина**:

Основні терміни

Контейнер - це виконуваний екземпляр, який інкапсулює необхідну програмне забезпечення. 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 програмного забезпечення, особливо в динамічних і розподілених середовищах та керування додатками.

Dockerfile — файл-інструкція по збірці образу. Дані не мають розширення (на зразок .txt).

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 Engine — це движок Docker, власне, це сам докер.

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 програмного забезпечення. Для офіційних образів перераховані їх потенційні уразливості. Ця інформація відкрита для будь-якого зареєстрованого користувача. Доступні як безкоштовні, так і платні акаунти.

Docker — це програмне забезпечення, яке дає можливість на певній ділянці пам'яті ізольовано встановити необхідну ОС (операційну систему), версію Java, налаштувати змінні оточення, встановити різні залежності і дати доступ тільки за певних умов. При цьому дану програму абсолютно не буде хвилювати, що відбувається навколо.

Тепер простими словами: це можливість, образно кажучи, відділити собі кімнату, поклеїти там шпалери, які вам подобаються, розставити меблі і техніку до смаку, встановити замок на двері і ключ видати тільки братові. При цьому, якщо ви зайдете в іншу кімнату, там все буде по-іншому і ніяких проблем не буде.

Контейнери докорінно змінюють спосіб розробки, поширення і функціонування програмного забезпечення. Розробники можуть створювати програмне забезпечення у локальній системі, яке буде працювати однаково у будь-якому операційному середовищі - в програмному комплексі ІТ-відділу або на ноутбуці користувача, або в хмарному кластері. Масштаби переходу у застосуванні контейнерів в інформаційних технологіях стрімко зростають при розробці і впровадженні програмного забезпечення у різних сферах промисловості.

Контейнери (containers) являють собою засоби інкапсуляції додатків разом з його залежностями (поєднання в одному місці додатка і залежностей).

Тобто контейнер - це стандартна одиниця програмного забезпечення, в яку упаковано додаток з усіма необхідними для його роботи залежностями - кодом програми, середовищем запуску, системними інструментами, бібліотеками і налаштуваннями. Контейнер не залежить від ресурсів або архітектури хоста, на якому він працює.

Віртуальна машина (ВМ) — емулятор апаратного забезпечення. Простими словами на своєму комп'ютері за допомогою програми ви можете встановити собі ще якусь ОС (операційну систему), яку зможете запустити паралельно.

Є певні відмінності між ВМ і контейнеризацією. Для простоти розуміння, технологія контейнеризації дозволяє запустити дуже компактні віртуальні машини (окремі процеси). Такі контейнери займають мало місця (образ Ubuntu займає 68Mb), запускаються дуже швидко і налаштовуються дуже гнучко.

Принципові відмінності щодо використання віртуальних машин і контейнерів полягають у наступному:

- метою застосування віртуальної машини є повна емуляція іншого програмного (операційного) середовища;

- метою застосування контейнера є зробити додатки переносимими, ізольованими, самодостатніми.

Переваги контейнерів:

- гнучке середовище, контейнери можна створювати значно швидше, ніж екземпляри віртуальних машин, їх розмір невеликий

- вимірюється в мегабайтах, на відміну від розмірів віртуальної машини, яка містить операційну систему, розмір якої може бути декілька гігабайтів; крім того, контейнери швидко запускаються;

- підвищена продуктивність, кожний контейнер можна розглядати як окремий мікросервіс, його оновлення не потребує синхронізації;

- управління версіями, дозволяє відстежувати версії контейнера та їх відмінності;

- переносимість середовища обчислення, контейнери інкапсулюють операційні системи та залежності додатків, тобто легко переносити образ контейнера з одного середовища в інше, наприклад, Windows/Linux;

- стандартизація, більшість контейнерів засновані на відкритих стандартах і можуть працювати в усіх основних дистрибутивах Windows/Linux;

- безпека, контейнери ізолюють процеси одного контейнера від іншого, тобто будь-яке оновлення в одному контейнері не впливає на інший.

До недоліків контейнерів можна відзначити наступні:

- підвищена складність, якщо працює n-контейнерів з додатком, це збільшує коефіцієнт складності, а управління множиною контейнерів може бути складною задачею в умовах виробництва;

- більшість контейнерних технологій, таких як Docker, засновані на Linux-контейнерах (LXC), тому їх запуск у середовищі Microsoft може викликати певні складнощі;

- незрілість, контейнери – це відносно нова технологія на ринку, тому для при виникненні проблем необхідно витратити певний час для їх вирішення.

Класифікація контейнерів

Для створення контейнерів операційних систем використовуються наступні технології:

- LXC (Linux Container);

- Open VZ, технологія базується на ядрі ОС Linux і дозволяє на одному фізичному сервері створювати і запускати ізольовані один від одного копії обраної операційної системи (Debian, CentOS, Ubuntu);

- Linux VServer (Linux Virtual Server), технологія дозволяє створювати декілька віртуальних серверів, які працюють незалежно під управлінням одного ядра операційної системи;

- BSD Jailis, технологія віртуалізації в системі FreeBSD, що дозволяє створювати всередині однієї операційної системи FreeBSD кілька незалежно працюючих FreeBSD на тому ж ядрі операційної системи, але абсолютно незалежно налаштовуються з незалежним набором встановлених додатків;

- Solaris Container (Solaris Zone), технологія, що дозволяє розділити на програмному рівні Solaris 10 ОС на контейнери (зони), майже самостійні ОС, які можуть мати окремі незалежні ресурси (процесори, пам'ять, дисковий простір) і своїх власних користувачів.

Для створення контейнерів для додатків на різних платформах Windows, Mac, Linux Ubuntu використовується контейнерна технологія Docker, а для операційної системи CoreOS на базі Linux контейнерна технологія Rocket. Це власний контейнерний движок Rocket (rkt – keep reading), який дозволяє працювати з публічними хмарними додатками.

Загальна схема роботи Docker

Схема створення контейнера виглядає наступним чином:



1. Створюєте 'Dockerfile' — файл, в якому необхідно описати, як буде створюватися образ. Простими словами — це опис того, як буде виглядати ваша кімната;
2. Image — це образ, на підставі якого в подальшому буде запущений контейнер. Це дизайнерський проект вашої кімнати, чітка схема того, що і де буде стояти;
3. Container — це запущений образ, в якому працює Ваша програма з описаними залежностями відповідно до інструкції. Тобто, це вже готова кімната, в якій ви можете жити.

**Завдання**

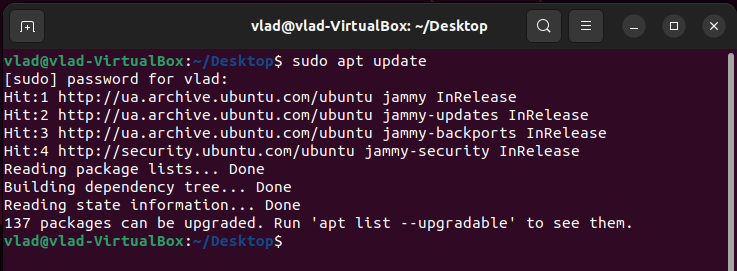
1. Ознайомитися з теоретичними матеріалом по лабораторній роботі.

2. Опанувати команди, які використовують при установці Docker та його встановити.

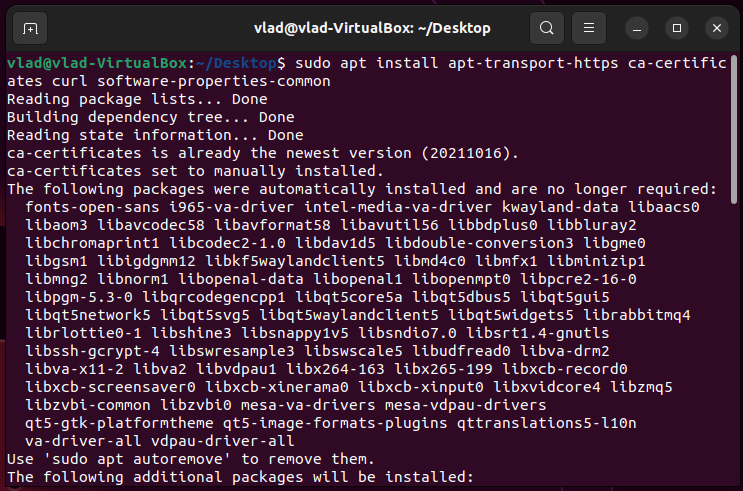
3. Підготувати звіт з описом процесу установки Docker з наведенням screenshot-ів екрану при виконанні кожної дії, надати його для викладача.

**Результат виконання роботи**

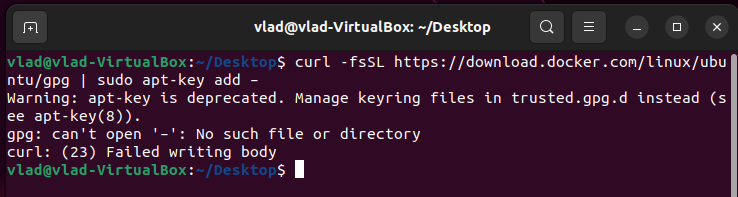
Спочатку оновлюємо існуючий перелік пакетів за допомогою команди *sudo apt update*:

****

Далі встановлюємо необхідні пакети, за допомогою команди *sudo apt install apt-transport-https ca-certificates curl software-properties-common*, які дозволяють менеджеру пакетів apt використовувати пакети по HTTPS :

****

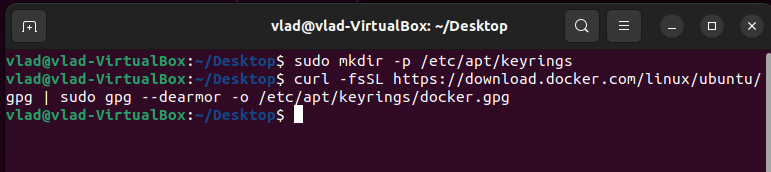
Потім додаємо в свою систему ключ GPG офіційного репозиторію Docker:

****

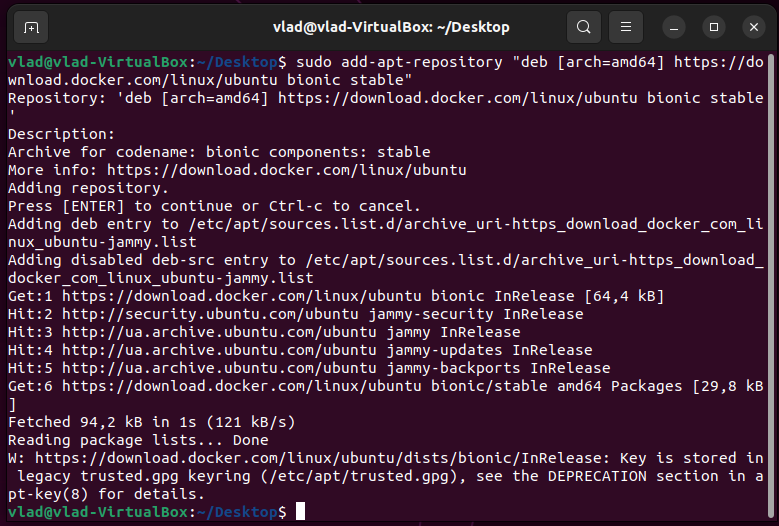
Через те що apt-key застарілий, нас просять використовувати trusted.gpg.d. Тому перейдемо на основний сайт Docker та знайдемо інструкції скачування Docker та виконаємо пункт додавання в свою систему ключа GPG офіційного репорзиторію Docker.

****

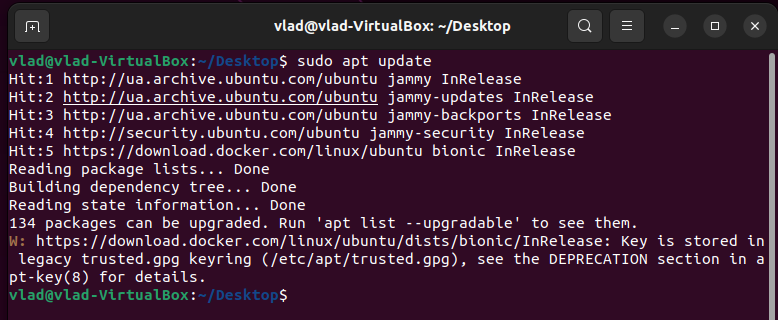
Виконаємо наведені команди у консолі

****

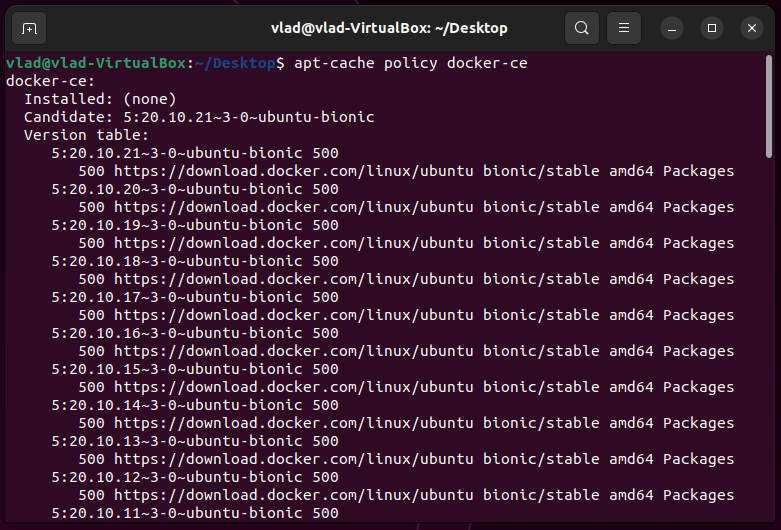
Додаємо репозиторій Docker в список джерел пакетів APT за допомогою команди *sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"*

****

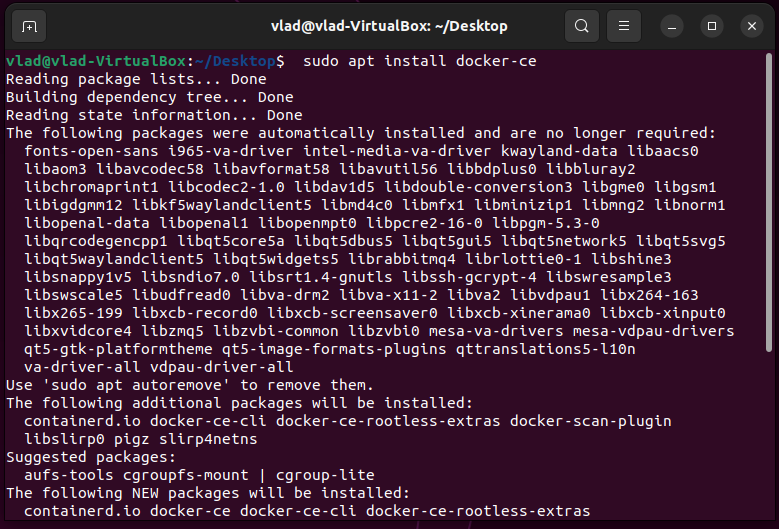
Оновлюємо базу даних пакетів інформацією про пакети Docker зі знову доданого сховища за допомогою команди *sudo apt update*:

****

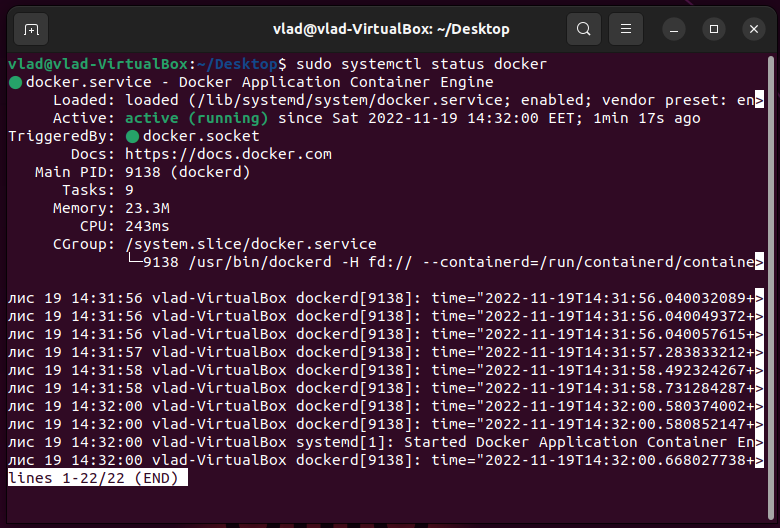
Слід переконатися, що ми встановлюємо Docker з репозиторію Docker, а не з репозиторію за замовчуванням Ubuntu, ввівши *apt-cache policy docker-ce*:

****

Далі встановлюємо Docker ввівши команду *sudo apt install docker-ce*:

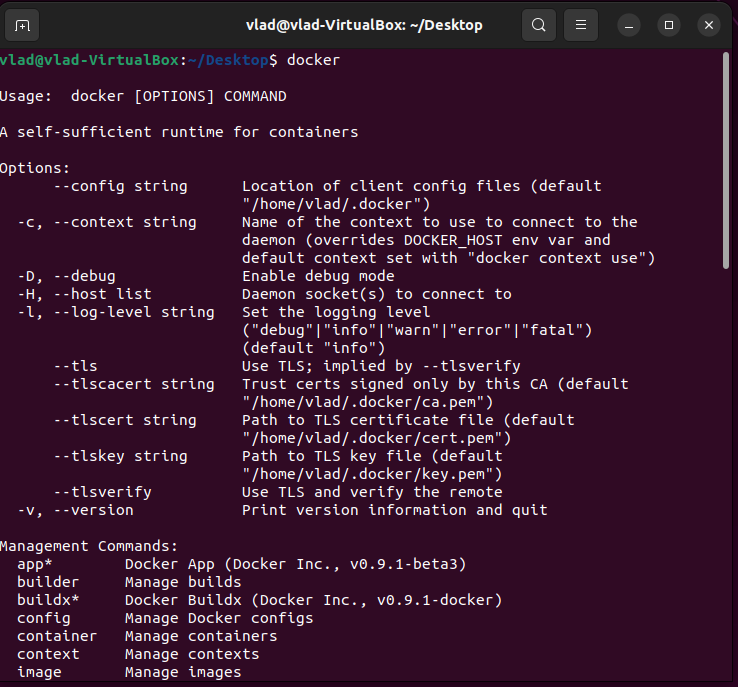
****

Docker встановлений, демон запущений, і процес буде запускатися при завантаженні системи. Переконаємося, що процес запущений ввівши команду sudo systemctl status docker:

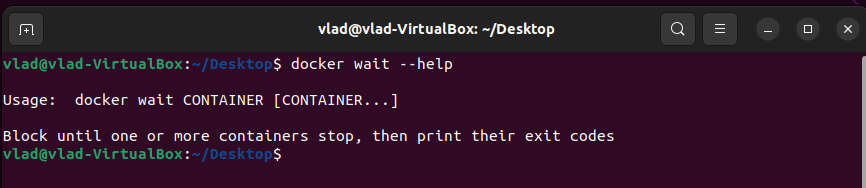
****

**Використання команди Docker**

Команда docker дозволяє використовувати різні опції, команди з аргументами. Синтаксис команди наступний: docker [option] [command] [arguments]. Для перегляду усіх опцій, доступних команд управління та підкоманд, вводимо команду *docker*

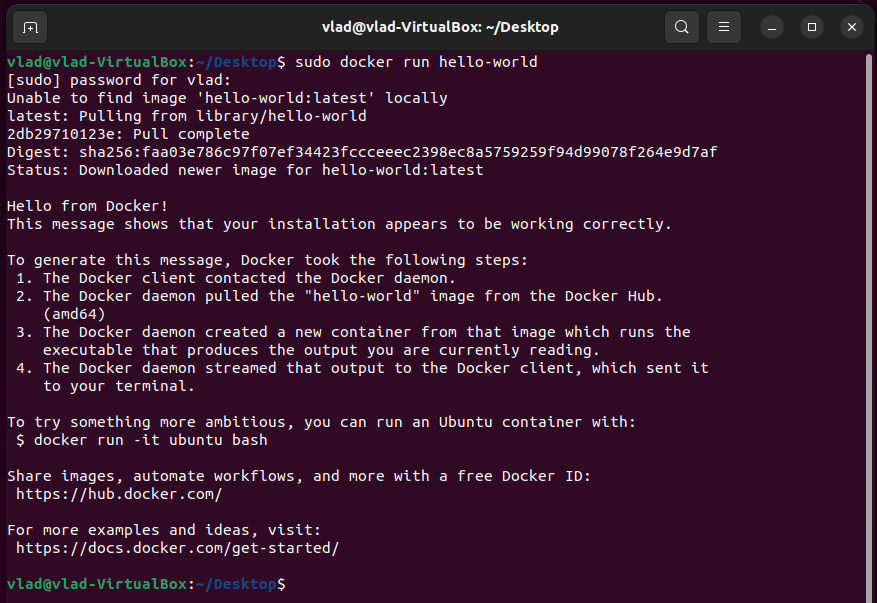
****

Для перегляду опцій використання певної команди вводимо docker docker-subcommand –help. Наприклад переглянемо опцію використання команди wait, ввівши команду *docker wait --help*.

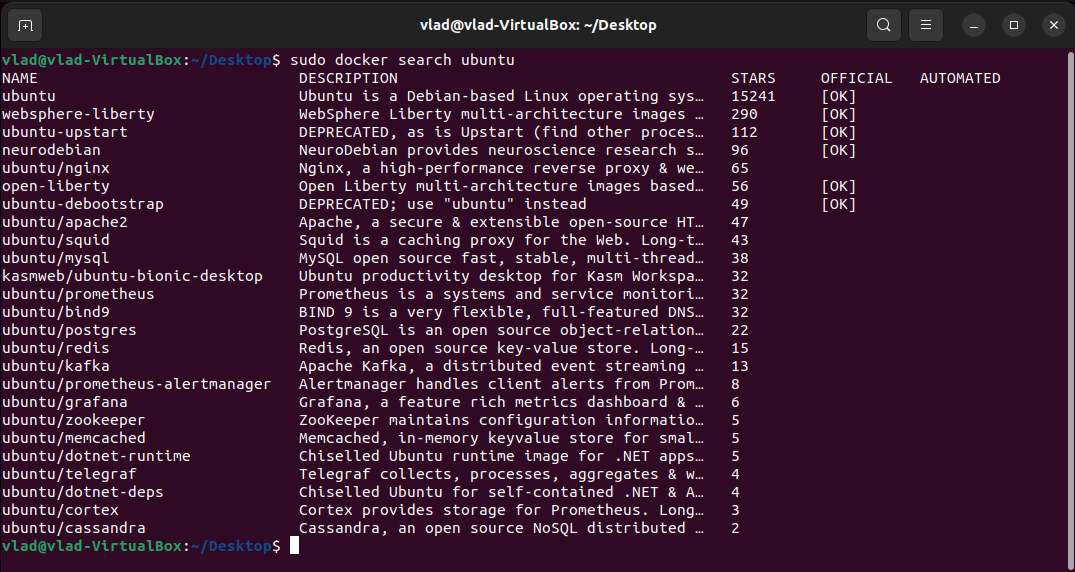
****

**Робота з образами Docker**

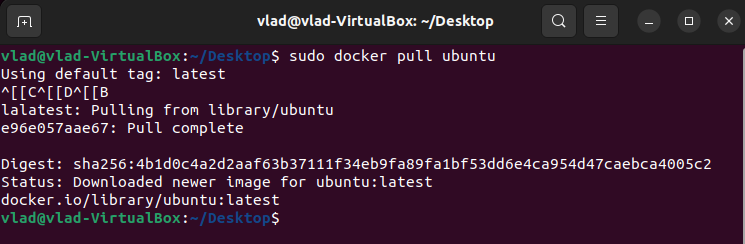
Контейнери Docker запускаються з образів Docker. За замовчуванням Docker отримує образи з хаба Docker Hub, який являє собою реєстр образів і він підтримується компанією Docker. Будь-хто може створити і завантажити свої образи Docker в Docker Hub, тому для більшості додатків і дистрибутивів Linux, які можуть знадобитися вам для роботи, вже є відповідні образи в Docker Hub. Для перевірки чи є доступ до образів і чи є можливість завантажувати образи з Docker Hub, вводимо команду *sudo docker run hello-world*. Спочатку Docker не міг знаходити образ hello-world локально, тому завантажував образ з Docker Hub, який є репозиторієм за замовчуванням. Після завантаження образу Docker створював з образу контейнер і запускав додаток в контейнері, відображаючи повідомлення.

****

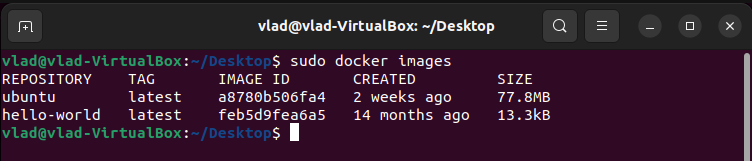
Образи, доступні в Docker Hub, можна шукати за допомогою команди docker search. Наприклад, для пошуку образу Ubuntu вводимо *sudo docker search ubuntu*. Скрипт переглядає Docker Hub і повертає список всіх образів, імена яких підходять під заданий пошук.

****

Коли потрібний образ обраний, можна завантажити його на комп'ютер за допомогою підкоманди pull. Щоб завантажити офіційний образ ubuntu на комп'ютер, використовуємо команду *sudo docker pull ubuntu*

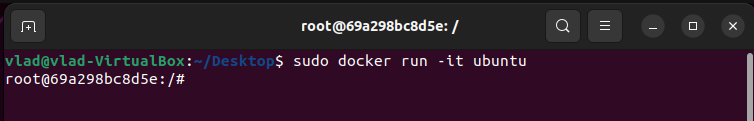
****

Для перегляду завантажених на комп'ютер образів потрібно ввести *sudo docker images*

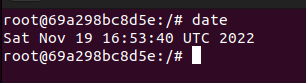
****

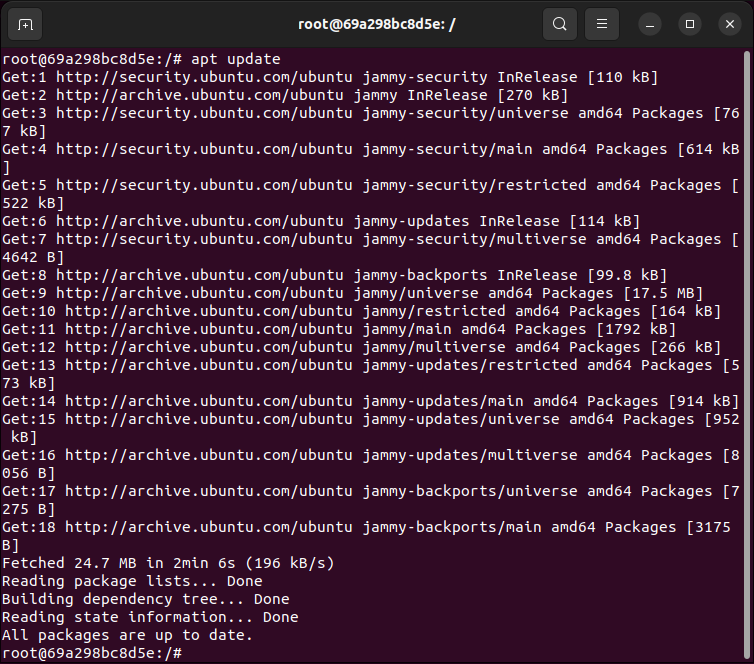
**Запуск контейнера Docker**

Комбінація параметрів -i та -t забезпечує інтерактивний доступ до командного процесора контейнера:

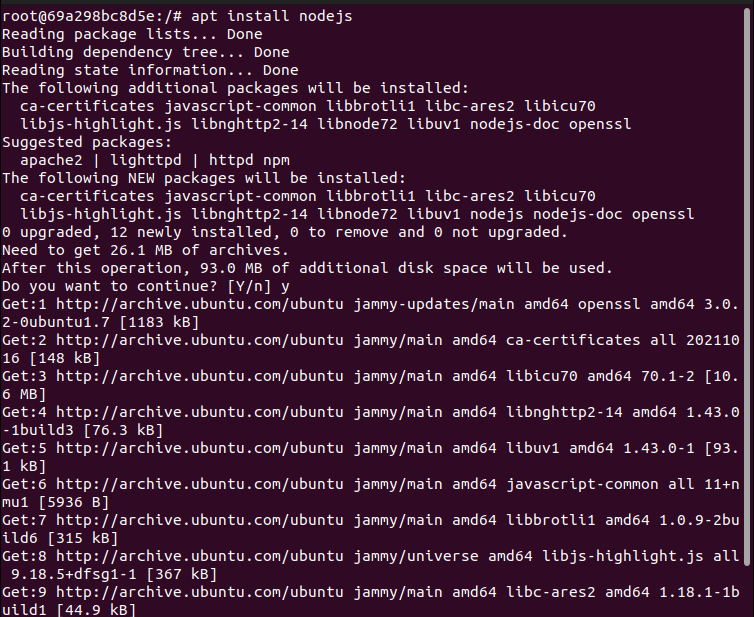
****

Тепер можна запускати будь-які команди всередині контейнера. Наприклад, вивести дату(*date*) або оновити базу даних пакета всередині контейнера(*apt update*). Перед командами не потрібно використовувати sudo, оскільки ви працюєте всередині контейнера як користувач з привілеями root

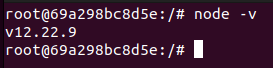
****

****

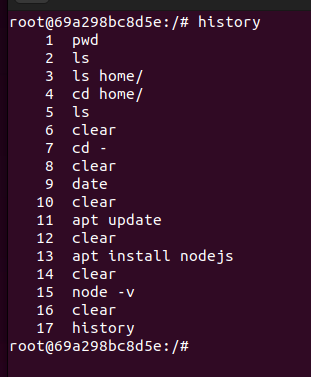
Тепер в контейнері можна встановити будь-який додаток. Спробуємо встановити Node.js (середовище виконання JavaScript):

****

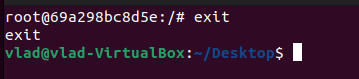
Ця команда встановлює Node.js в контейнер з офіційного репозиторію Ubuntu. Коли установка завершена, переконаємося, що Node.js встановлений

****

Також можемо вивести історію команду у самому контейнері

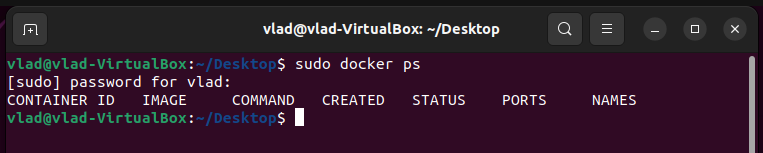
****

Всі зміни, які здійснюються всередині контейнера, застосовуються лише для цього контейнера. Щоб вийти з контейнера, вводимо команду *exit*.

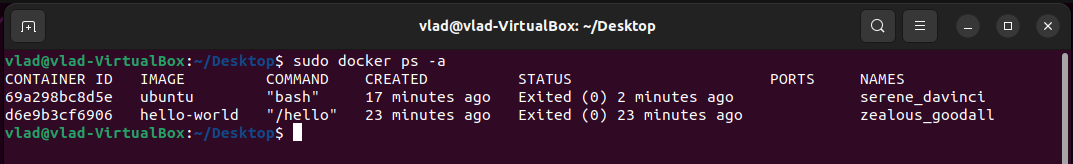
****

**Управління контейнерами Docker**

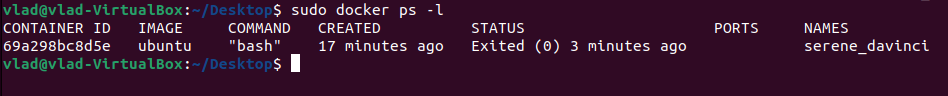
Для перегляд активних контейнерів використовуємо команду *sudo* *docker ps*

****

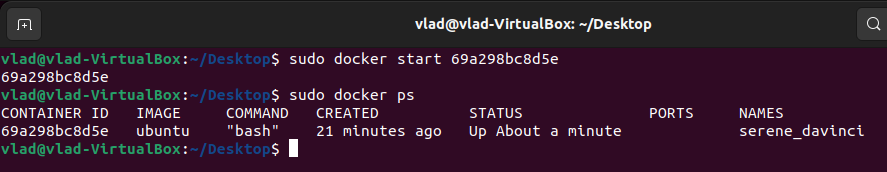
Щоб побачити усі контейнери, використовуємо параметр -a : *sudo docker ps -a*

****

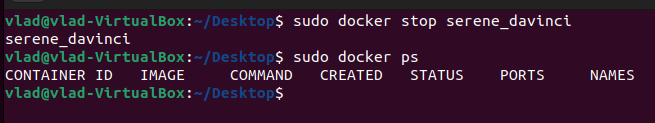
Для перегляду останнього створеного контейнерів, використовуємо параметр -l: *sudo docker ps -l*

****

Для запуску зупиненого контейнера використовуємо команду *docker start*, а потім вказуємо ідентифікатор контейнера або його ім'я. Наприклад, запустимо контейнер ubuntu та введемо його ID, ввівши команду *sudo docker start 69a298bc8d5e*. Також введемо команду *sudo docker ps* щоб переглянути активні контейнери.

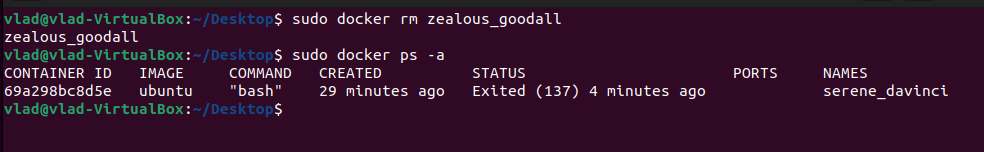
****

Для зупинки запущеного контейнера використовуємо команду docker stop, потім вказуємо ідентифікатор контейнера або його ім'я. Цього разу ми використовуємо ім'я, яке призначив контейнеру Docker, тобто serene\_davinci і введемо команду *sudo docker stop serene\_davinci*. Та виведемо активні контейнери ввівши команду *sudo docker ps*.

****

Якщо вам контейнер більше не потрібен, видаляємо його командою docker rm із зазначенням або ідентифікатора, або імені контейнера.

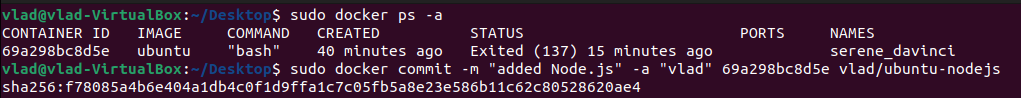
Для того аби видалити непотрібний контейнер, використовується команда docker rm та вказується ID або ім’я контейнера. Для прикладу видалимо контейнер hello-world ввівши команду *sudo docker rm zealous\_goodall* де zealous\_goodall це імя контейнера, яке автоматично йому присвоєне

****

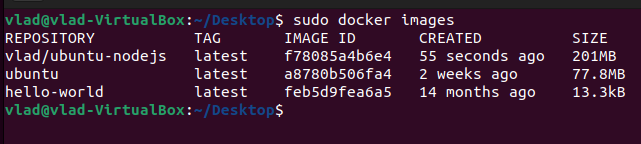
**Збереження змін в контейнері в образ Docker**

Для збереження змін в новому образі Docker використовується наступна команда : *docker commit -m "What you did to the image" -a "Author Name" container\_id repository /new\_image\_name*

Параметр «-m» дозволяє задати повідомлення про підтвердження, щоб полегшити користувачам образу розуміння того, які зміни були внесені, а параметр «-a» дає змогу вказати автора.

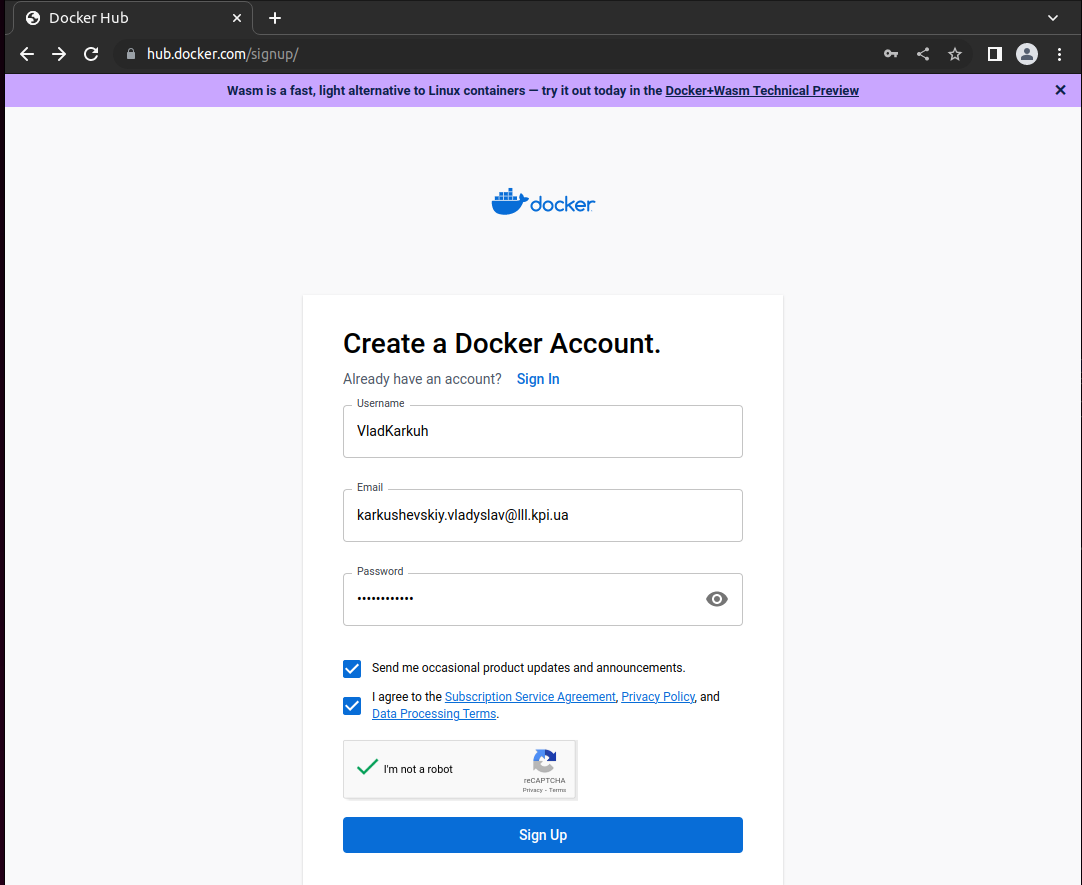
****

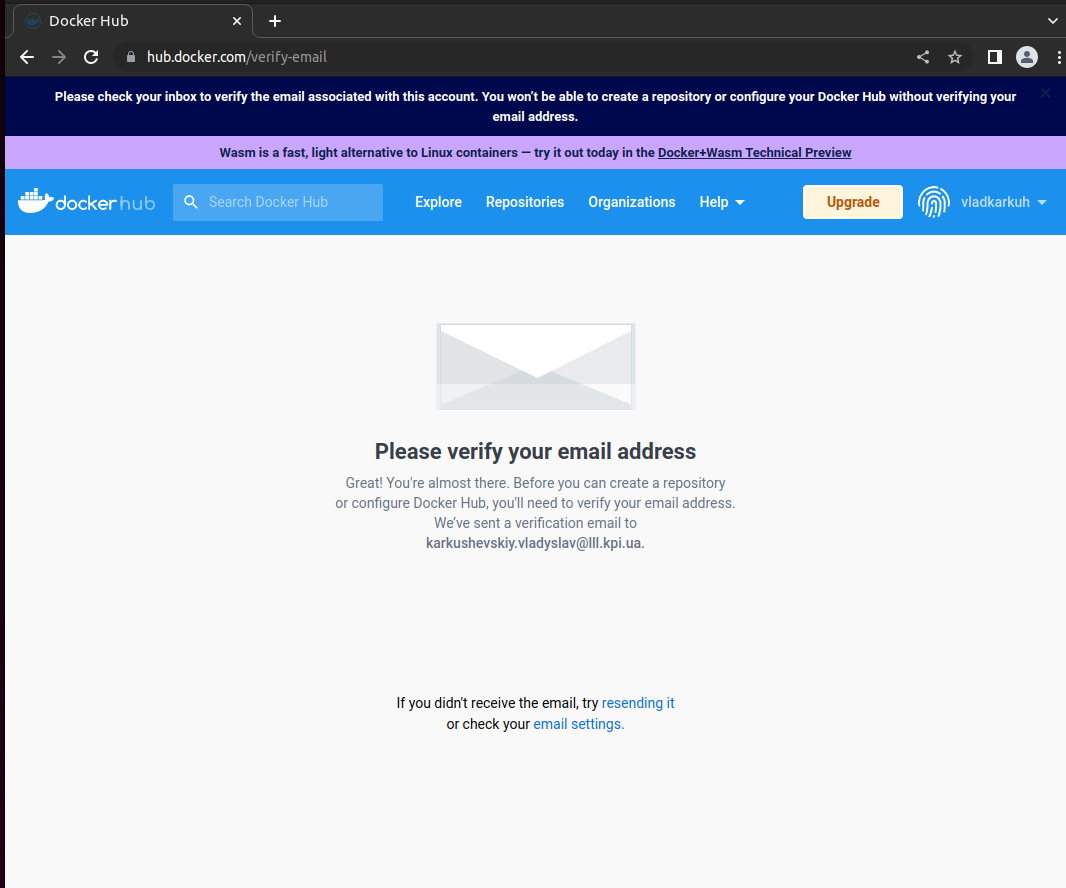
Також виведемо завантажені контейнери, і побачимо що зявився новий.

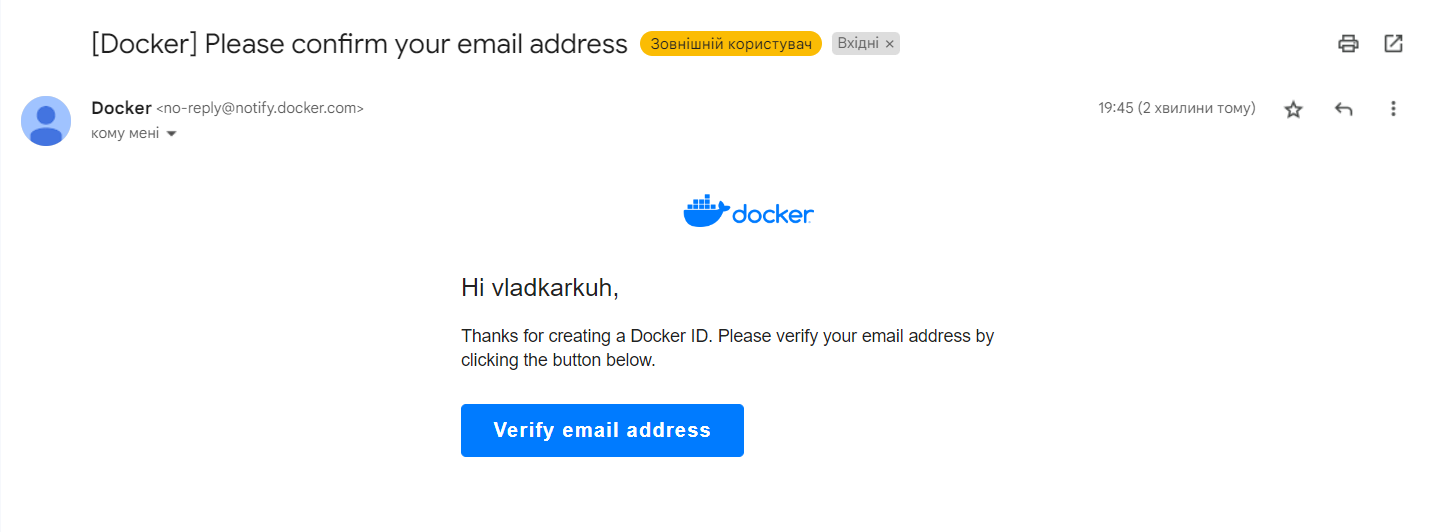
****

**Завантаження контейнерів Docker в репозиторій Docker**

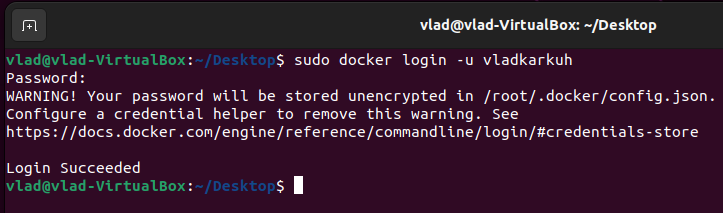
Для початку, потрібно зареєструватися на DockerHub. Перейдемо на офіційний сайт та зареєструємся

****

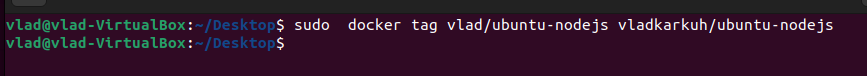
****

****

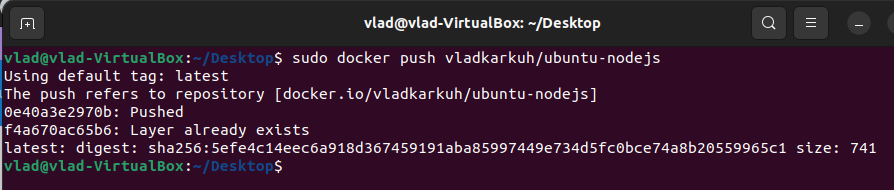
Тепер потрібно увійти у DockerHub , тому введемо команду *sudo docker login -u vladkarkuh*

****

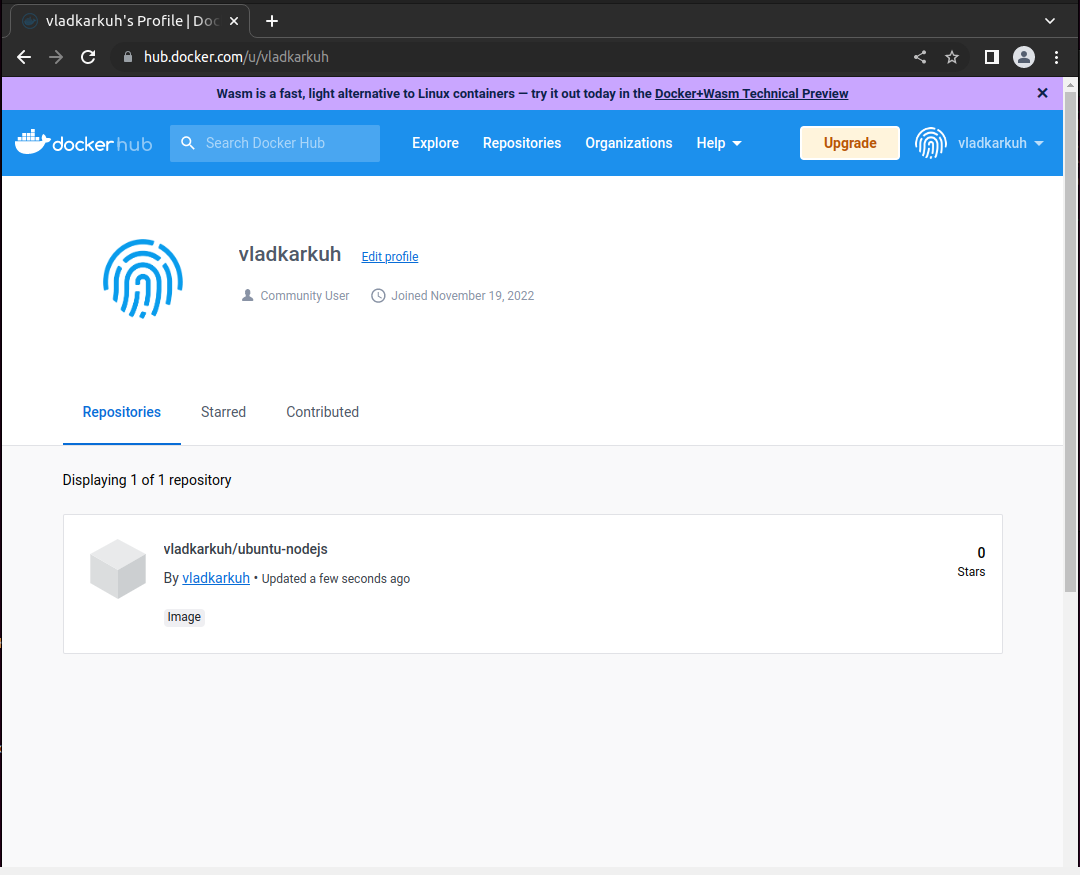
Якщо ім'я користувача в Docker-реєстрі відрізняється від локального імені користувача, яке використовувалося для створення образу, необхідно прив'язати свій образ до імені користувача в реєстрі ввівши команду docker tag username/ubuntu-nodejs docker-registry- /ubuntu-nodejs

****

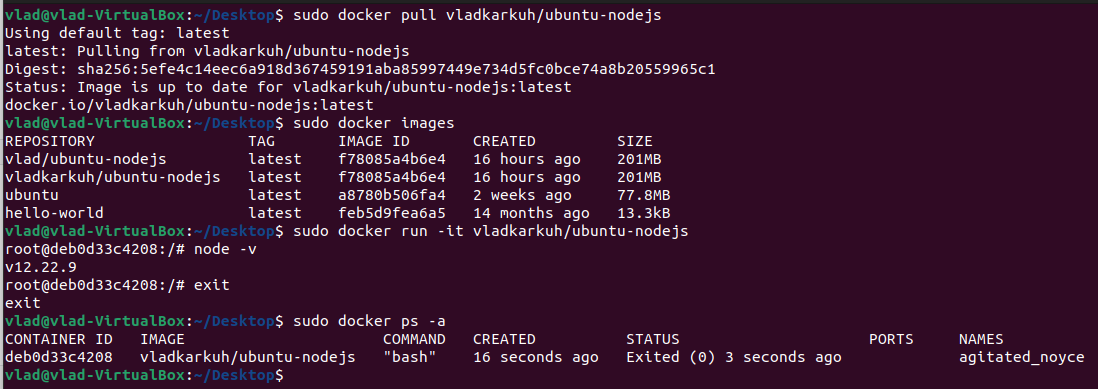
Далі завантажуємо власний образ Docker Hub використавши команду sudo docler push docker-registry-username/docker-image-name



Після виконання команди push у DockerHub з’являється контейнер



Далі за допомогою команди *sudo docker pull vladkarkuh/ubuntu-nodejs* можна завантажити образ на нову машину і використовувати його для запуску нового контейнера

****

**Висновок:**

На цій лабораторній роботі було набуто навичок встановлення додатку Docker в ОС Linux. Ознайомлено з теоретичним матеріалом по лабораторній роботі. Опановано команди які використовувалися при встановленні Docker та встановлено його. Створенно акаунт на DockerHub та завантажено туди контейнер.