Міністерство освіти й науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Кафедра автоматизації проектування енергетичних процесів і систем

ЗВІТ

Про виконання лабораторної роботи №8

З дисципліни «Архітектура системного програмного забезпечення»

**Установка Docker**

TI-92 Черноусова Дениса

Перевірила д.т.н. Левченко Л.О.

Київ - 2021

**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 програмне забезпечення з відкритим кодом, принцип роботи якого найпростіше порівняти з транспортними контейнерами. Філософію Docker часто описують за допомогою метафори «доставки універсальних вантажних контейнерів», тобто стандартизованих розмірів контейнерів, які можна переміщувати між різними видами транспорту (вантажівками, поїздами, кораблями) з мінімумом ручної праці. 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.

Саме контейнери Docker спрощують перенесення програмних додатків.

***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 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.

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

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 не віртуалізацію, а 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.

На відміну від контейнера, віртуальна машина (ВМ) працює під управлінням повноцінної операційної системи, включаючи її власне ядро, і є повною емуляцією іншого програмного (операційного) середовища. Перевагами та метою створення контейнерів є прискорення розробки, інкапсуляція додатків (залежностей додатків, операційних систем) та переносимість програмного забезпечення.

Всі контейнери створюються з образів контейнерів. Образи контейнерів представляють собою набір файлів, організованих в стек шарів, розташованих на локальному комп'ютері або у віддаленому реєстрі контейнерів. Образ контейнера складається з файлів операційної системи 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.

Таким чином, Docker – це стандартизоване пакетне програмне забезпечення, призначене для розробки, розгортання проектів та використання розроблених додатків, які є переносимими та самодостатніми, у той час як метою віртуальної машини є емуляція іншого операційного середовища. Docker дозволяє відокремити ваш додаток від вашої інфраструктури і дозволяє запустити буд-який додаток, який безпечно ізольований у контейнері. Docker має особливі образи програмного забезпечення, що запускаються у віртуальному середовищі, не створюючи повну копію ОС). Слід зазначити, що розробники можуть одночасно запускати десятки контейнерів, що дає можливість імітувати роботу промислової розподіленої мережі.

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.

Docker доступний для будь-якої з операційних систем: Windows, Linux, Maс OS. Docker ставиться на версію Ubuntu 18.04, на Ubuntu 19.10 не ставиться. Docker дозволяє запустити ОС Linux в ізольованому середовищі дуже швидко, протягом декількох хвилин. Платформа Docker складається з двох окремих компонентів:

* *Docker Engine*, механізму, що відповідає за створення і функціонування контейнерів,
* *Docker Hub,* хмарного сервісу для поширення контейнерів.

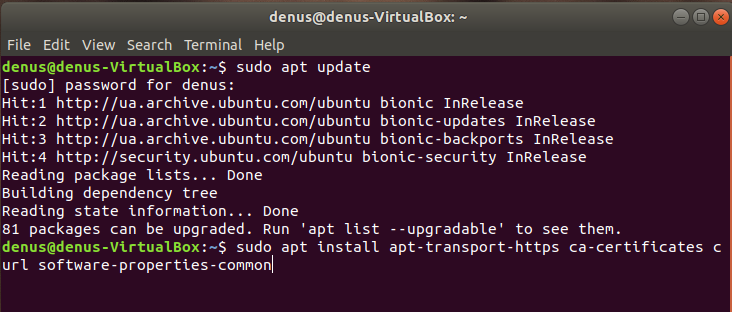
Механізм *Docker Engine* надає ефективний і зручний інтерфейс для запуску контейнерів. До цього для запуску контейнерів, що використовують таку технологію, як, наприклад, LXC (Linux Container), були потрібні неабиякий запас спеціальних знань в цій області і великий обсяг ручної роботи. *Docker Hub* надає величезну кількість образів контейнерів з відкритим доступом для завантаження, дозволяючи користувачам швидко почати роботу з ними і уникнути рутинної роботи, раніше вже виконану іншими людьми.

Трохи пізніше були розроблені ***інструментальні засоби для Docker***:

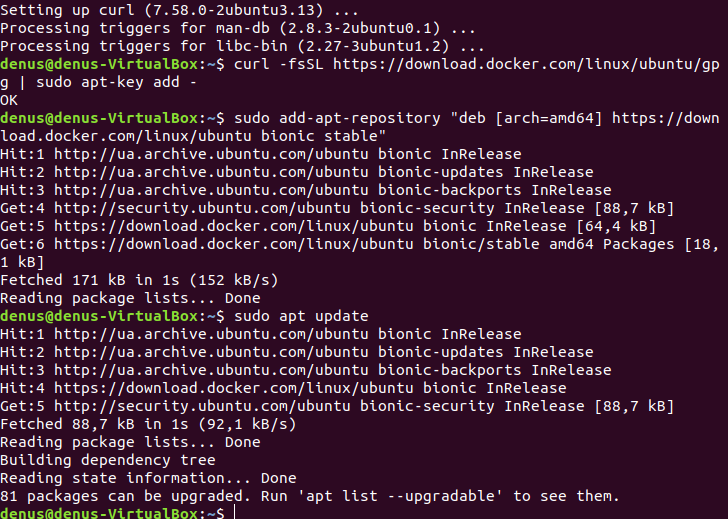
* ***Swarm*** - менеджер кластерів,
* ***Kinematic*** - графічний користувальницький інтерфейс для роботи з контейнерами;
* ***Machine*** - утиліта командного рядка для підтримки роботи Docker-хостів.

**Завдання :**

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

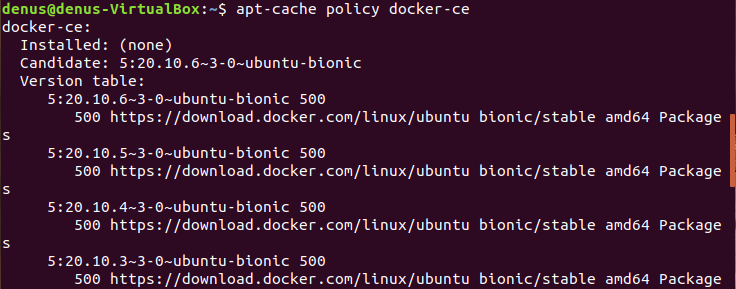
**Хід виконання роботи**

Оновлення системи та встановлення необхідних пакетів, які дозволяють менеджеру пакетів apt виконувати пакети HTTPS

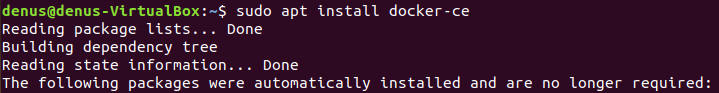
Додаємо в свою систему ключ GPG офіційного репозиторію Docker.

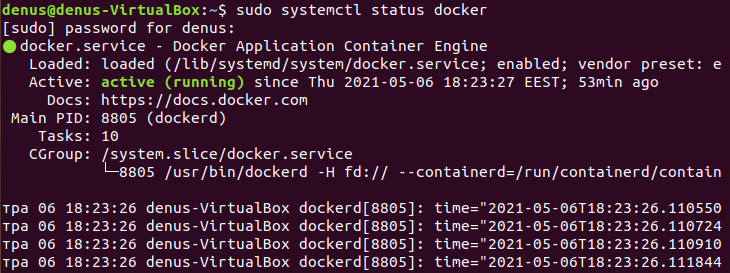
Додаємо репозиторій Docker в список джерел пакетів APT

Оновлюємо базу даних пакетів інформацією про пакети Docker

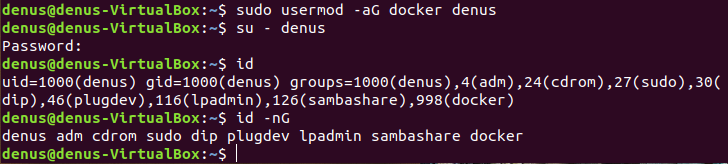


Дізнаємося звідки скачаємо Docker

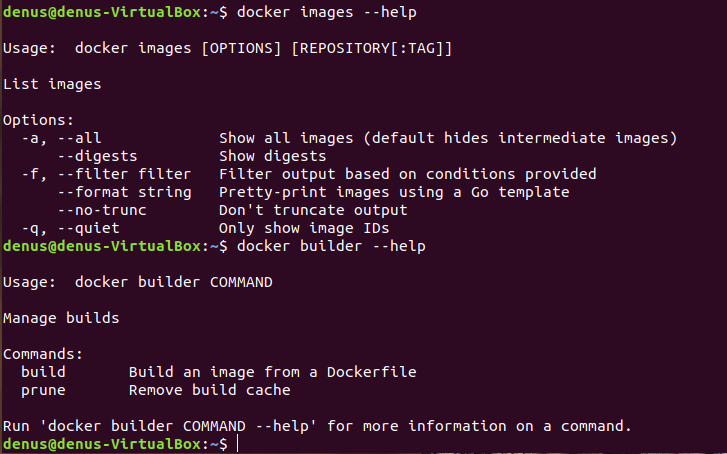
Скачання докеру

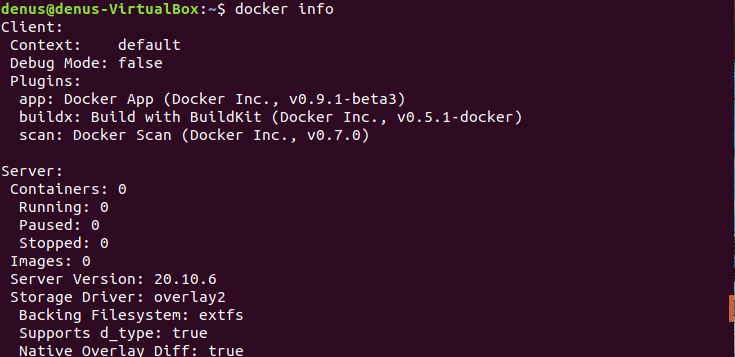


Активний докер

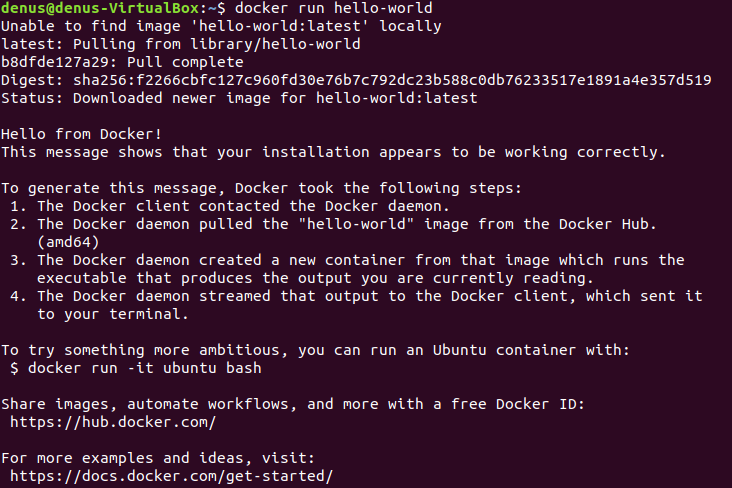


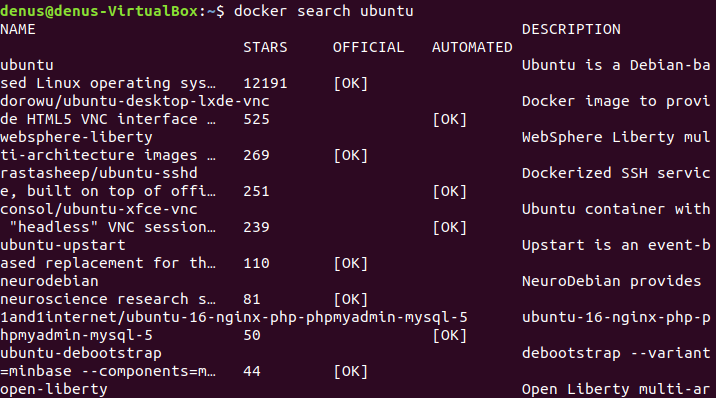
Додаємо свого користувача до групи docker з необхідними правами

Використання команди і ключа help

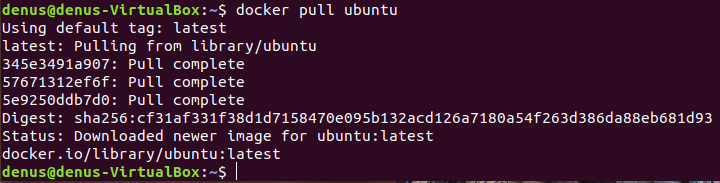


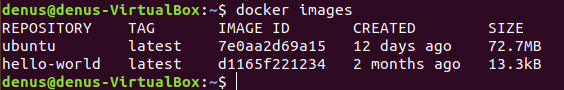
Виведення інформації про контейнери та образи

Завантаження та запуск образу з Docker Hub

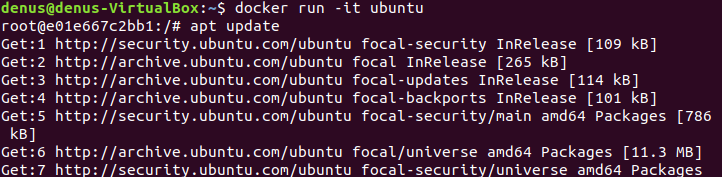


Пошук образу в Docker Hub

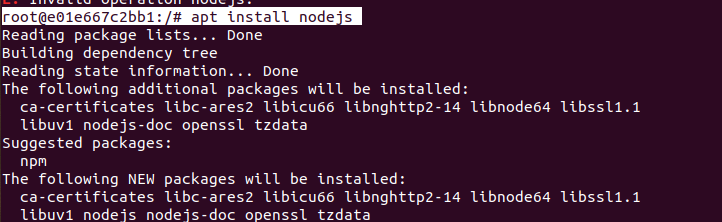
Скачування образу на локальний комп’ютер



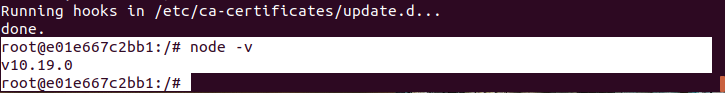
Виведення усіх образів, що на комп’ютері



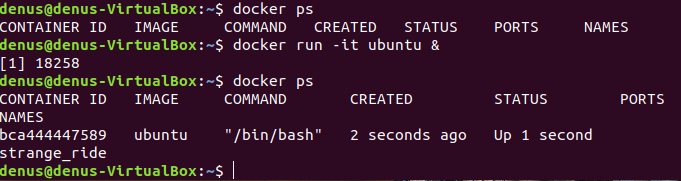
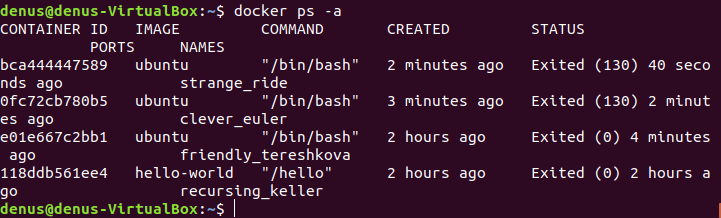
Запуск в інтерактивному режимі контейнеру за образом ubuntu



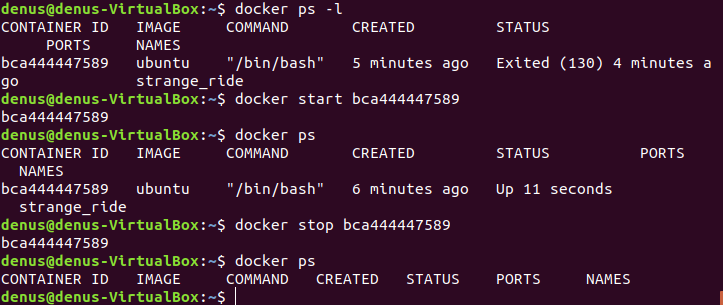
Встановлення компілятора всередині контейнеру

Виведення версії компілятора

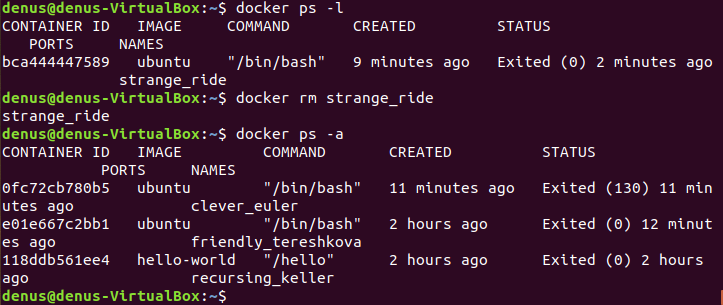
Вихід з контейнеру

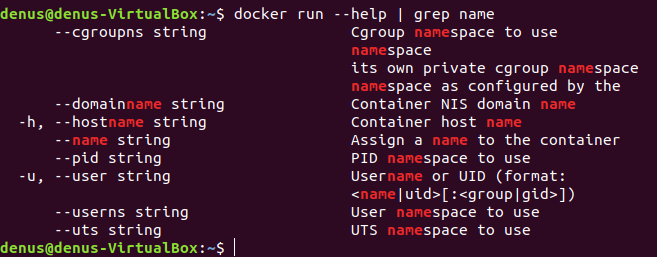
Відображення усіх працюючих контейнерів

Відображення усіх контейнерів

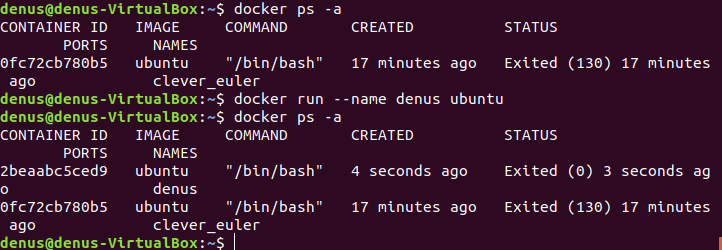


Відображення останнього контейнера. Старт та зупинка його

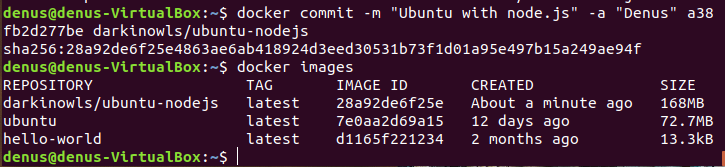
По іменне видалення контейнеру

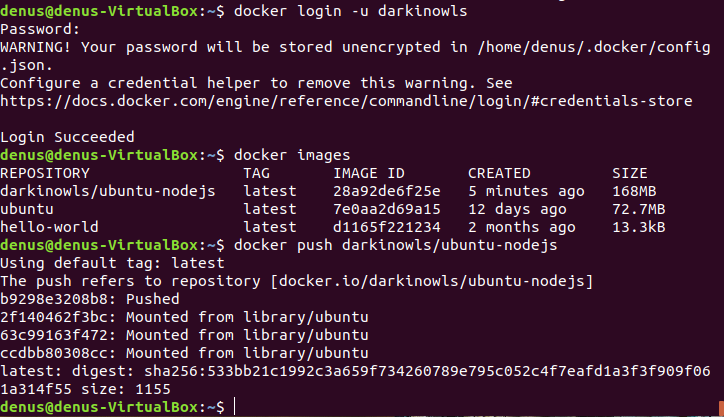


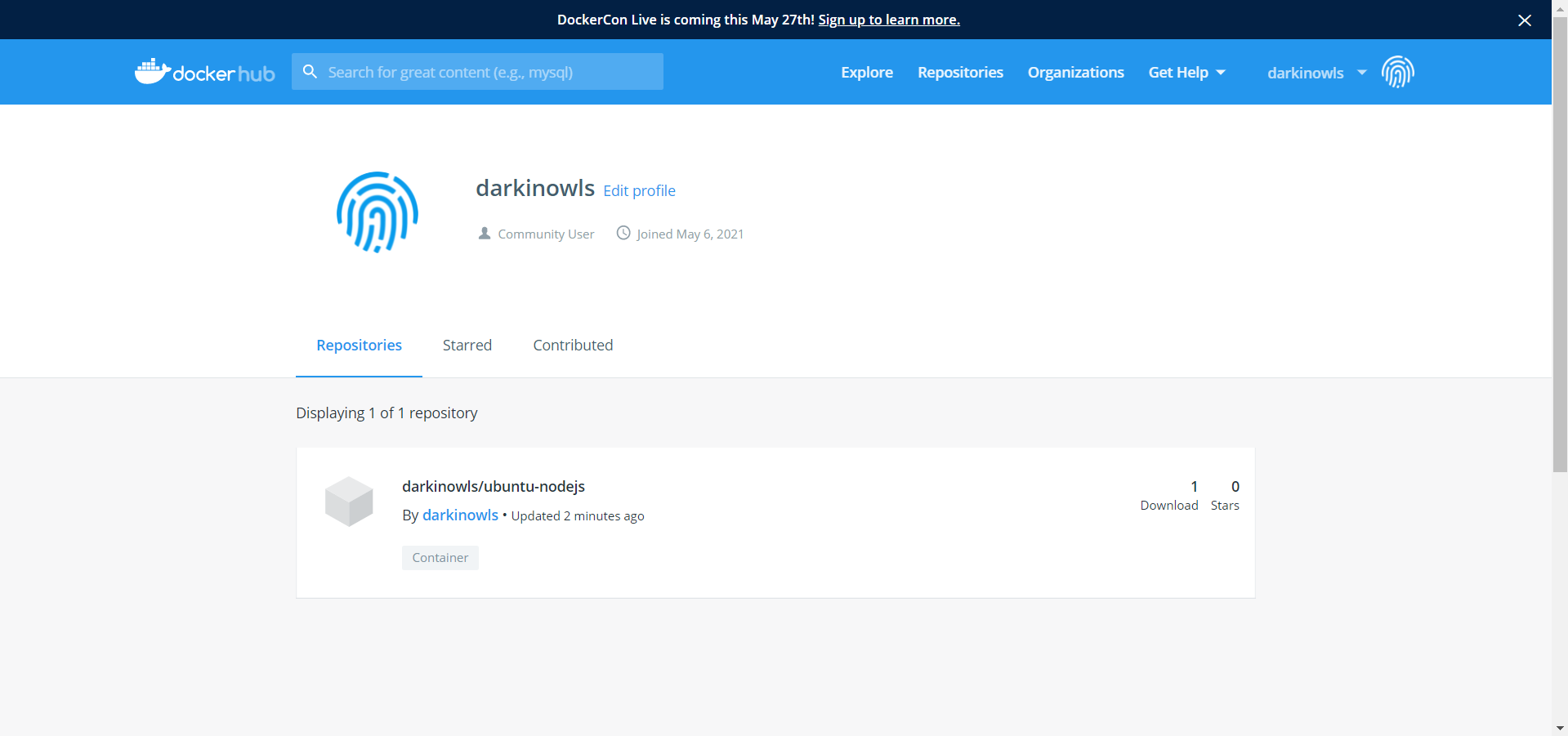
Знаходження потрібного ключа



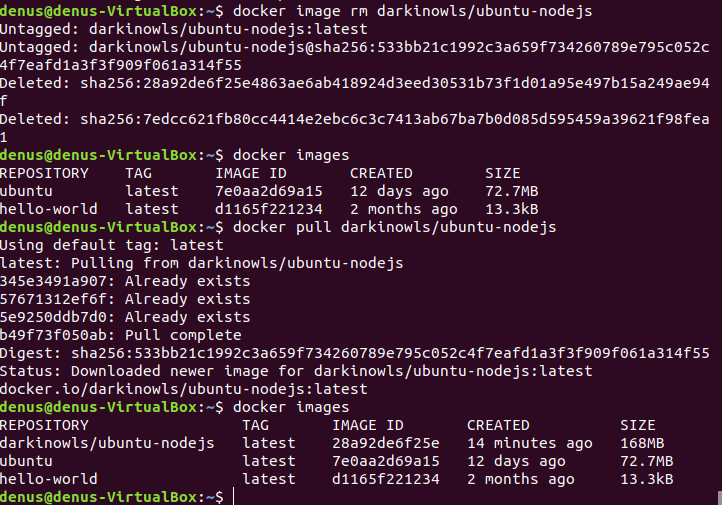
Створення нового контейнеру з ключем name

Зберігання контейнеру



Авторизація та відправка збереженого контейнеру як образу на Docker Hub

Образ на Docker Hub

Видалення образу з локального комп’ютеру та скачування образу з свого репозиторію

**Висновок**

Під час роботи над лабораторною роботою були здобуті навички встановлення додатка Docker в ОС Linux та роботи з контейнерами, образами. Був створений контейнер Ubuntu, на якому інстальоване середовище виконання JavaScript – Nodejs. Новостворений образ був збережений в реєстрі на особистому акаунті Docker Hub.