ЗВІТ

з лабораторної роботи

З дисципліни «Розгортання інформаційно-комунікаційних систем»

Студента групи ІК-41: Симчук К.Ю.

Тема роботи: Перші контейнери Linux Alpine

Мета роботи: запустити популярний, безкоштовний, легкий контейнер та дослідити основи роботи контейнерів, як Docker Engine виконує та ізолює контейнери один від одного, запулити своє зображення

Теоретичні відомості

Віртуальна машина - це апаратна абстракція: вона забирає фізичні центральні процесори та оперативну пам'ять від хоста, а також розподіляє їх і ділить між кількома меншими віртуальними машинами. Всередині віртуальної машини працює ОС і додаток, але програмне забезпечення для віртуалізації зазвичай не знає про це.

Контейнер - це абстракція програми : основна увага приділяється ОС та додатку, а не стільки апаратній абстракції. Сьогодні багато клієнтів фактично використовують як віртуальні машини, так і контейнери у своїх середовищах, і фактично можуть запускати контейнери всередині віртуальних машин.

Alpine - це легкий дистрибутив Linux, тому його можна швидко завантажити та запустити, що робить його популярною відправною точкою для багатьох інших зображень.

Команда pull вибирає альпійське **зображення** з **реєстру Докера** і зберігає його в нашій системі. У цьому випадку реєстр - це [Docker Hub](https://hub.docker.com/).

Команда docker image дозволяє побачити список усіх образів у вашій системі.

Принцип роботи команд

Під час виклику run клієнт Docker знаходить зображення (у цьому випадку alpine), створює контейнер, а потім запускає команду в цьому контейнері. Під час запуску docker container run alpineви надали команду ( ls -l), тому Docker виконав цю команду всередині контейнера. Після виконання lsкоманди контейнер закрився.

Контейнери Docker функціонують на рівні додатка, тому вони пропускають більшість кроків, які вимагають віртуальні машини, і просто запускають те, що потрібно для програми.

Незважаючи на те, що кожна docker container runкоманда використовувала один і той же альпійський образ , кожне виконання було окремим ізольованим контейнером . Кожен контейнер має окрему файлову систему і працює в іншому просторі імен; за замовчуванням контейнер не має можливості взаємодіяти з іншими контейнерами, навіть з тими ж зображеннями.

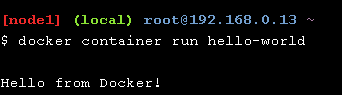
Термінологія

* Зображення - файлова система та конфігурація нашої програми, які використовуються для створення контейнерів. Щоб дізнатися більше про образ Docker, запустіть docker image inspect alpine. У демонстраційній версії вище ви використовували docker image pullкоманду для завантаження альпійського зображення. Коли ви виконували команду docker container run hello-world, вона також docker image pullза кадром завантажувала образ світу привіт .
* Контейнери - запущені екземпляри зображень Docker - контейнери запускають фактичні програми. Контейнер містить програму та всі її залежності. Він поділяє ядро ​​з іншими контейнерами та працює як ізольований процес у просторі користувача на ОС хосту. Ви створили контейнер, за допомогою docker run якого ви завантажили альпійське зображення. Список запущених контейнерів можна побачити за допомогою docker container lsкоманди.
* Демон Docker - фонова служба, що працює на хості, яка керує створенням, запуском та розповсюдженням контейнерів Docker.
* Клієнт Docker - інструмент командного рядка, що дозволяє користувачеві взаємодіяти з демоном Docker.
* Docker Hub - Магазин, серед іншого, є [реєстром](https://store.docker.com/) образів Docker. Ви можете розглядати реєстр як каталог усіх доступних образів Docker. Ви будете використовувати це пізніше в цьому підручнику.

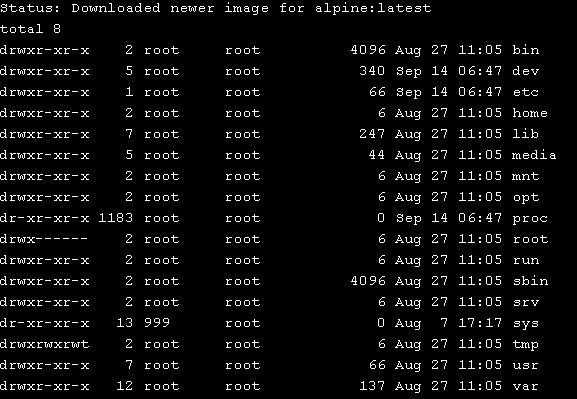
Хід виконання роботи

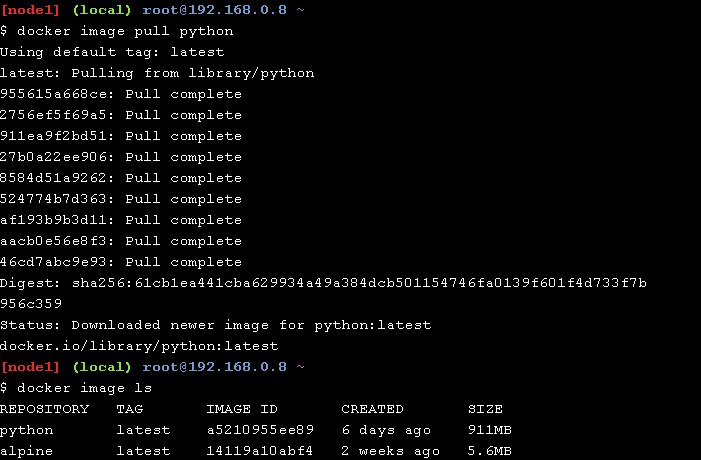
В цій роботі було виконано кілька пунктів:

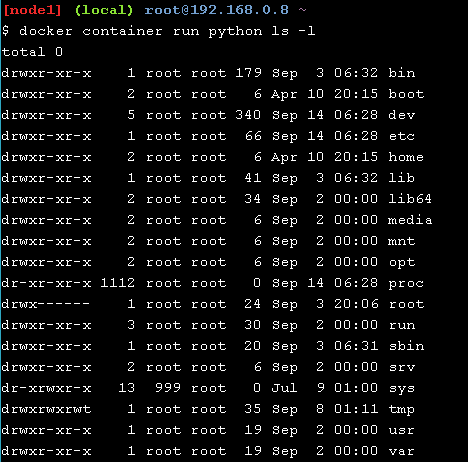
1. Командою docker container run hello-world запустили свій перший контейнер. Образ цей береться з реєстру Docke, так як локально збережених зображень в нас немає.

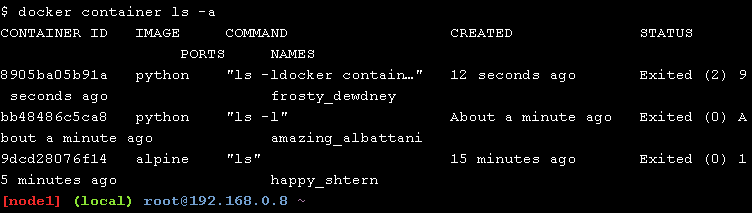


1. Запулили образ alpine командою pull, далі командою docker container run alpine ls –l запускаємо контейнер, де бачимо список всіх каталогів контейнеру.



1. Виконала різні команди в різних екземплярах контейнера.
2. Розібралась, що таке ізоляція контейнерів , те що екземпляри контейнерів не мають можливість взаємодіяти з іншими.
3. Запулила інший образ python: 
4. Подивилась список його каталогів





Посилання на репозиторій: https://github.com/katyasymchuk/labDoker.git

Висновок : на даній лабораторній роботі я ознайомилась з роботою контейнерів, з командами для роботи з ними. Розібралась, що ізоляція потрібна не тільки для безпеки, але і для тестування наслідків внесення змін до програми. Запулила свій образ Python, який був в реєстрі Doker.