Технічні науки

УДК 004.054

Приходько Владислав Сергійович
студент
Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського»
Приходько Владислав Сергеевич
студент
Национального технического университета Украины
«Киевский политехнический институт имени Игоря Сикорского»
Prykhodko Vladyslav
Student of the
National technical university of Ukraine
«Igor Sikorsky Kyiv Polytechnic Institute»

Чанцова Катерина Вікторівна
студент
Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського»
Чанцова Катерина Викторовна
студент
Национального технического университета Украины
«Киевский политехнический институт имени Игоря Сикорского»
Chantsova Kateryna
Student of the
National technical university of Ukraine
«Igor Sikorsky Kyiv Polytechnic Institute»

ОРГАНІЗАЦІЯ І ПЕРЕВАГИ ПЛАТФОРМИ DOCKER
ОРГАНИЗАЦИЯ И ПРЕИМУЩЕСТВА ПЛАТФОРМЫ DOCKER
THE ORGANIZATION AND ADVANTAGES OF THE DOCKER PLATFORM

Анотація. Розгляд архітектури платформи Docker і її переваги над іншими платформами.

Ключові слова: Docker Engine, Docker REST API, архітектура платформи, компоненти платформи, переваги платформи.

Аннотация. Рассмотрение архитектуры платформы Docker и ее преимущества перед другими платформами.

Ключевые слова: Docker Engine, Docker REST API, архитектура платформы, компоненты платформы, преимущества платформы.

Summary. Consider the architecture of the Docker platform and its advantages over other platforms.

Key words: Docker Engine, Docker REST API, platform architecture, platform components, platform advantages.

Docker відкрита платформа для розробки, публікації, і запуску додатків. Дана платформа дає можливість відокремити програми від інфраструктури, щоб можливо було швидко зробити розгортання. З Docker можливо керувати інфраструктурою так само просто, як і додатками.

Докер забезпечує можливість упаковки і запуску додатку в ізольованому середовищі званим контейнером. Ізоляція і необхідний рівень безпеки дозволяють запускати безліч контейнерів на заданому хості. Через легкий характер контейнерів, які працюють без додаткового навантаження гіпервізора, можна запустити більше контейнерів, ніж на віртуальній машині [1].

Сама платформа називається Docker Engine, яка представляє клієнт-серверний додаток з трьома головними компонентами (рис. 1):

- Сервер працює у фоновому режимі (демон).
- REST API, який використовують програми для взаємодії з сервером.
- Інтерфейс командного рядка (CLI) клієнт.

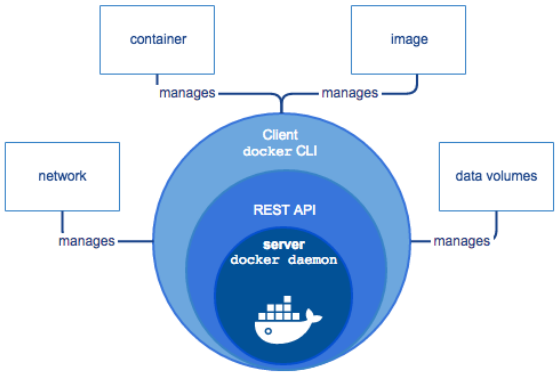

Рис. 1. Склад платформи Docker [2]

CLI використовує Docker REST API для керування або взаємодії з демоном Докер за допомогою сценаріїв або безпосередньо команд CLI. Багато інших додатків Docker засновані на використанні API і CLI.

Демон створює і управляє об'єктами Docker, такими як образи, контейнери, мережі та сховища даних.

1. Архітектура платформи Docker Engine

Docker використовує архітектуру клієнт-сервер (рис. 2). Docker клієнт спілкується з демоном Docker, який бере на себе створення, запуск, розподіл контейнерів. Обидва, клієнт і сервер можуть працювати на одній системі, також можна підключити клієнт до віддаленого демона docker. Клієнт і сервер спілкуються через сокет або через RESTful API [2].

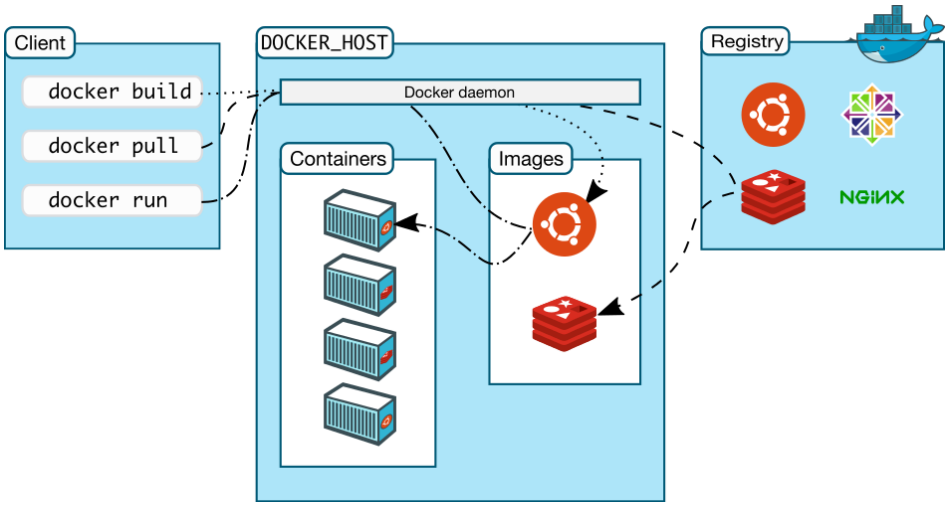

Рис. 2. Архітектура платформи Docker [2]

Як показано на рис. 2, демон запускається на хост-машині. Користувач не взаємодіє з сервером на пряму, а використовує для цього клієнт. Docker-клієнт - головний інтерфейс до Docker системи. Він отримує команди від користувача і взаємодіє з docker-демоном. Демон Docker працює на хост-машині. Користувач використовує клієнт Docker для взаємодії з демоном.

2. Головні компоненти платформи Docker Engine

Щоб розуміти, як працює docker, потрібно знати про три основні його компоненти:

- образи (images)
- реєстр (registries)
- контейнери

Docker-образ - це read-only шаблон з набором інструкцій для створення контейнера. Наприклад, образ може містити операційну систему Ubuntu з веб-сервером NGINX і додатком на ній. Образи використовуються для створення контейнерів. Docker дозволяє створювати нові образи, оновлювати існуючі, або завантажувати і використовувати образи, які були створені іншими користувачами.

Реєстр - це бібліотека образів. Він може бути публічним або приватним і може розташовуватися на одному сервері з демоном, клієнтом або на окремому сервері. Реєстри - це компонента поширення.

Контейнери схожі на директорії. У контейнерах міститься все, що потрібно для роботи програми. Кожен контейнер створюється з образу. Контейнери можуть бути створені, запущені, зупинені або видалені. Кожен контейнер ізольований і є безпечною платформою для додатка. Контейнери - це компонента роботи [3].

Docker сервіс надає режим swarm, за допомогою якого можна обмежувати кількість примірників процесів образу. Можна вказати кількість паралельних завдань реплік для запуску, і менеджер swarm гарантує, що навантаження розподілиться рівномірно між робочими вузлами.

3. Переваги контейнеризації

Контейнери несуть в собі багато привабливих переваг як для розробників, так і для системних адміністраторів.

Деякі з найбільш привабливих переваг перераховані нижче.

1) Абстрагування хост-системи від контейнеризованих додатків. Контейнери були задуманими повністю стандартизованими. Це означає, що контейнер з'єднується з хостом або чим-небудь зовнішнім по відношенню до нього, за допомогою певних стандартизованих інтерфейсів. Контейнеризований додаток не повинен покладатися або якимось чином залежати від ресурсів або архітектури хоста, на якому він працює.
2) Простота масштабування. Одним з переваг абстрагування між операційною системою хоста і контейнерами є те, що при правильному проектуванні програми, масштабування може бути простим і прямолінійним. Сервіс-орієнтована архітектура в

комбінації з контейнеризованими додатками забезпечує основу для легкого масштабування.

3) Простота управління залежностями і версіями додатка. Контейнери дозволяють розробнику програми або компоненту додатку, з усіма його залежностями і далі працювати з ними як з єдиним цілим. Хосту не треба турбуватися про залежності, необхідні для запуску конкретного додатку. Якщо хост може запустити Docker, він може запустити будь-який Docker-контейнер.
4) Надзвичайно легкі, ізольовані середовища виконання. Контейнери ізольовані на рівні процесів, працюючи при цьому поверх одного і того ж ядра хосту. Це означає, що контейнер не включає в себе повну операційну систему, що призводить до практично миттєвого його запуску.
5) Спільно використовувані шари. Контейнери легкі ще і в тому сенсі, що вони зберігаються "пошарово". Якщо кілька контейнерів засновані на одному і тому ж шарі, вони можуть спільно використовувати цей базовий шар без дублювання, що призводить до мінімального завантаження дискового простору в наступних образах.
6) Можливість компонування та передбачуваність. Docker-файли дозволяють користувачам задати конкретні дії, необхідні для створення нового образу контейнера. Це дозволяє задавати налаштування середовища виконання так, як нібито це код, при бажанні, зберігаючи ці налаштування в системі контролю версій. Однакові Docker-файл, зібрані в одному і тому ж оточенні, завжди створюють ідентичні образи контейнеру [4].

Docker базується на технологіях namespaces і cgroups (перша забезпечує ізоляцію, друга - угруповання процесів і обмеження ресурсів),

тому в плані віртуалізації він мало чим відрізняється від звичних нам LXC / OpenVZ. Та ж швидкість роботи, ті ж методи ізоляції, засновані на механізмах ядра Linux, але він надає зручне API для взаємодії з ними, що дозволяє розгорнути повноцінне віртуальне оточення і запустити в ньому додаток так само просто, як, наприклад, перезапустити веб-сервер.

Література

1. Знакомимся с основными возможностями Docker [Электронный ресурс] – Режим доступа: https://xakep.ru/2015/06/01/docker-usage/ – Дата доступу: 24.07.2017.
2. Офіційний сайт документації Docker [Електронний ресурс] – Режим доступу: https://docs.docker.com/ – Дата доступа: 24.07.2017.
3. Введение в Docker [Электронный ресурс] – Режим доступа: http://docker.cool/docs/docker-engine/docker-overview/ – Дата доступа: 24.07.2017.
4. Виртуализация процесса разработки [Электронный ресурс] – Режим доступа: https://dou.ua/lenta/articles/docker/ – Дата доступа: 24.07.2017.