Інформаційні технології

УДК 004.051

Паріс Сергій Павлович

студент

Інституту прикладного системного аналізу

Національного технічного університету України

«Київський політехнічний інститут імені Ігоря Сікорського»

АВТОМАТИЗОВАНИЙ ІНСТРУМЕНТАРІЙ РОЗГОРТАННЯ ХМАРНИХ СЕРВІСІВ

Анотація. *У статті розглянуті існуючі системи для автоматизації розгортання сервісів*

Ключові слова: *хмарні сервіси, розгортання сервісів, Kubernetes, Docker, Docker Swarm, Mesos, Rancher.*

Актуальність теми. З розвитком інтернету та технологій розробки підхід до розробки та надання послуг кардинально змінився. Сьогодні у інтернеті зберігається велика кількість веб-сайтів та різноманітної інформації. Разом із розвитком інтернету розвивались і веб-сервіси та способи їх розміщення на серверах. Спочатку це були власні сервери компаній, які мали безліч недоліків починаючи від потреби зберігати власний штат співробітників і закінчуючи тим, що ресурси кожного із серверів не використовувались на 100%, пізніше з'явились віртуальні машини, які теж мали свої недоліки у виді потреби віртуалізувати для кожної віртуальної машини операційну систему та фізичні компоненти комп'ютера. Ще пізніше з'явились контейнери, які використовуються на даний момент у світі. Відносно недавно кількість контейнерів, які розгортає компанія зросли до сотень а інколи і тисяч і гостро постало питання автоматичного

розгортання таких сервісів. Саме цю проблему і вирішують такі системи як Kubernetes. Вони дозволяють автоматично розгортати безліч мікросервісів налаштувавши їх один раз конфігураційними файлами, а також підтримувати їх роботу.

Docker - це технологія віртуалізації контейнерів. Отже, це як дуже легка віртуальна машина - VM. Це додаток для побудови контейнерів, робота якого насправді полягає в допомозі людям у створенні контейнерів та додатків всередині ізольованого простору, а також зберігання копій цих контейнерів для того, щоб потім ділитися ними серед своїх товаришів по команді і без проблем розгортати такі копії як в хмарі, так і на локальному комп'ютері розробника для аналізу, наприклад, якщо виникли якісь проблеми.

Є кілька проблем, які Docker вирішує. Перша із них про те, що VM є досить великим обчислювальним ресурсом. Середня віртуальна машина - це копія операційної системи, яка працює поверх гіпервізора – програмного забезпечення, що дозволяє паралельно керувати декількома операційними системами, що запущені поверх іншої операційної системи, що працює поверх фізичного обладнання, над яким потім знаходиться ваша програма. Це представляє певні виклики щодо швидкості та продуктивності, а також деякі проблеми у спритному середовищі, а також проблеми с швидкодією під час розгортання та зупинки роботи.

Kubernetes - це механізм організації контейнерів (COE) із відкритим кодом, натхненний проектом Google під назвою Borg. Kubernetes використовується для організації груп контейнерів, що представляють екземпляри програм, які відокремлені від машин, на яких вони працюють. Оскільки кількість контейнерів у кластері збільшується до сотень або тисяч екземплярів, а компоненти додатків розгортаються як окремі контейнери, Kubernetes приходить на допомогу, забезпечуючи основу для розгортання,

управління, автоматичного масштабування, високої доступності та відповідних завдань.

Контейнери - це хороший спосіб об'єднати та запустити ваші програми. У виробничому середовищі вам потрібно керувати контейнерами, в яких запущені програми, і переконатися, що немає простоїв. Наприклад, якщо контейнер опускається, потрібно запускати інший контейнер. Ось так на допомогу приходить Kubernetes. Kubernetes надає фреймворк для стійкого запуску розподілених систем. Він дбає про масштабування та відновлення після відмови для вашої програми, надає схеми розгортання тощо. Наприклад, Kubernetes може легко керувати розгортанням сервісів для вашої системи, або якщо був втрачений зв'язок із одним із контейнерів, або навіть комп'ютером, то Kubernetes знає, які контейнери і з якими параметрами були запущені на цьому комп'ютері і за лічені секунди запустить всі втрачені контейнери на інших комп'ютерах. Хоча дані із оперативної пам'яті тих контейнерів будуть втрачені, але кількість працюючих сервісів не буде зменшена і користувачу потрібно лише буде повторити свій виклик до сервісу аби отримати відповідь і весь цей процес можна автоматизувати і лише втручатись в нього для збільшення контейнерів або для вивчення результатів роботи системи за певний час для прийняття певних рішень.

Docker Swarm - це альтернативний, власний Docker механізм оркестрації контейнерів, який координує розміщення та управління контейнерами між декількома хостами Docker Engine. Docker Swarm дозволяє вам спілкуватися безпосередньо з множиною контейнерів, замість того, щоб спілкуватися з кожним Docker контейнером окремо. Архітектура Docker Swarm складається з двох типів вузлів, які називаються менеджерами та робітниками.

- Вузол - це машина, яка запускає екземпляр Docker Engine
- Swarm - це скупчення екземплярів Docker Engine.

- Вузол менеджера – вони розподіляють та планують вхідні завдання на вузли Worker та підтримують стан кластера. Вузли менеджера також можуть додатково запускати служби для вузлів Worker.
- Робочі вузли - це екземпляри Docker Engine, відповідальні за запуск програм у контейнерах.
- Сервіс - це образ мікросервісу, такого як Інтернет або сервери баз даних.

  Завдання - Служба, запланована для запуску на вузлі Worker.

Docker compose - це простий, але потужний інструмент, який використовується для запуску декількох контейнерів як однієї служби. Наприклад, припустимо, що є програма, яка вимагає Nginx як веб-сервера та PostgreSQL як служби баз даних. У цьому випадку за допомогою docker-compose можна створити один єдиний файл (docker-compose.yml), який створить обидва контейнери як одну службу, не запускаючи кожен окремо і корувати також можна як єдиним контейнером, дивлячись на одну консоль, в яку будуть виводити свої дані всі контейнери, разширити кількість контейнерів, або додати зовсім нові сервіси, також у створеної множини сервісів буде власна внутрішня мережа, та багато інших переваг.

Apache Mesos, загальнокластерний менеджер ресурсів, широко застосовується в кількох хмарах та центрах обробки даних. Mesos прагне забезпечити високе використання кластерів за допомогою тонкозернистого спільного планування ресурсів та справедливості ресурсів серед кількох користувачів завдяки розподілу на основі домінуючої справедливості ресурсів.

DRF враховує різні типи ресурсів (ЦП, пам'ять, дисковий ввід / вивід), що вимагаються кожною програмою, і визначає частку кожного ресурсу кластера, який може бути призначений додаткам. Месос прийняв дворівневу політику планування: DRF для розподілу ресурсів на конкуруючі рамки та планування рівня завдань кожною структурою для ресурсів, виділених на

попередньому кроці. Ми провели експерименти в локальному кластере Mesos, коли вони використовувались з такими фреймворками, як Apache Aurora, Marathon та наша власна фреймворкова Scylla, для вивчення справедливості ресурсів та використання кластера.

Mesos об'єднує всі ресурси в кластері та дозволяє чітко розподілити ресурси, дозволяючи та застосовуючи декілька додатків (званих Mesos framework) для спільного планування своїх завдань на віртуальних машинах / вузлах. Mesos використовує DRF для розподілу ресурсів у фреймворки, а потім фреймворки використовують алгоритми планування для планування завдань у межах виділених ресурсів.

Література

1. Diomidis S. Version Control Systems / Diomidis Spinellis, 2005. 2 p.
2. Mojtaba S. Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices / Mojtaba Shahin, 2017. 20 p.
3. Anderson C. Docker / Anderson Charles, 2015. 4 p.
4. Saha P. Exploring the Fairness and Resource Distribution in an Apache Mesos Environment / Saha Pankaj. Cloud and Big Data Laboratory, State University of, 2019. 8 p.
5. Stolberg S. Enabling Agile Testing Through Continuous Integration / Stolberg Sean, 2009. 6 p.