**Практична робота №11-2.** Kubernetes**.** (2 год.)

**Мета роботи**: ознайомлення з технологіями розподіленої обробки даних для обрання для розгортання програмного забезпечення

**Завдання:**

1. Ознайомитися з теоретичною частиною.
2. Надати письмові відповіді на запитання.
3. Прокоментувати переваги і недоліки використання технології Kubernetes.
4. Робота повинна бути виконана згідно критеріїв оформлення документації та повинна містити

* Назва практичної роботи.
* Прізвище, група
* Назва проекту.
* Заповнені таблиця та відповіді на запитання.

По закінченню лабораторну роботу потрібно здати на перевірку викладачеві, надіславши електронною поштою на адресу **[t.i.lumpova@gmail.com](mailto:t.i.lumpova@gmail.com)** . Якщо викладач знаходить помилки чи неточності, він може повернути роботу на доопрацювання.

Файл з роботою повинен мати назву в такому форматі:

**OPI<Номер групи><Номер лекції / практичної / лабораторної>[-<Номер завдання>][літера позначення типу роботи L – лекція, P –практична, R – лабораторна]<Прізвище англійською>**.. Наприклад, O**РІ3101Р**buts.doc.

Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності робіт -"близнюків" відповідь буде зараховуватися першому за часом надсилання.

**КОНТРОЛЬНІ ПИТАННЯ:**

1. В чому полягає віртуалізація?
2. В чому полягає контейнерізація?
3. В чому полягає різниця між віртуальною машиною та контейнером?
4. Визначте сильні сторони Kubernetes
5. Визначте слабкі сторони Kubernetes
6. Які програми отримають переваги від Kubernetes?
7. Навіщо потрібні контейнери?
8. Які завдання виконує Kubernetes?
9. Які функції виконує Docker?
10. Як ви розумієте оркестрацію?

**Теоретичні відомості**

Kubernetes (K8s) — це платформа з відкритим вихідним кодом, яка не залежить від постачальників і допомагає масштабувати, автоматично розгортати та керувати програмами, створеними за допомогою контейнерної технології. Kubernetes упорядковує контейнери вашої програми та забезпечує ефективне керування ними та відстеження. Спочатку розроблений групою інженерів Google і випущений у 2015 році Kubernetes став провідною системою оркестровки контейнерів. Kubernetes дозволяє командам керувати кількома контейнерами одночасно, заощаджуючи ваш час і зусилля.

Попередній підхід (до Kubernetes) до розробки програмного забезпечення полягав у тому, щоб об’єднати функції, логіку та різні сервіси, які складають програму, в одну програму. Однак ця монолітна архітектура не відповідала вимогам сучасної індустрії програмного забезпечення, які постійно змінювалися.

Оскільки програми з часом розширюються, може бути складно вносити точні зміни в код. Якщо його потрібно оновити, доведеться його повторно розгорнути, що призведе до простою та втрати грошей. Якщо в службі є помилка, її виправлення може залишити програму недоступною для користувачів. Масштабування однієї служби може конфліктувати з іншими.

Ці перешкоди усуваються за допомогою архітектури мікросервісів. За допомогою мікросервісів служби програми слабо пов’язані, і можна віртуалізувати їх за допомогою контейнерів. Розробник може самостійно кодувати, розгортати, оновлювати та підтримувати різні служби; також може збільшити або зменшити масштаб залежно від потреб своєї програми. Однак це означає, що в доведеться стежити за сотнями або навіть тисячами контейнерів.

Найпопулярнішим рішенням впорядкування цього божевілля є Kubernetes, який дає змогу командам керувати кількома контейнерами одночасно, заощаджуючи час і зусилля.

**Від віртуальних машин до контейнерів**

Раніше компанії розгортали програми на фізичному сервері з операційною системою та іншими розгорнутими програмами на додаток до цього. Якби програма вимагала більше ресурсів, інші програми не мали б того, що їм потрібно для коректної роботи.

Віртуальними машинами зручно керувати — зовсім не важливо, на якому залізі запущено програму, головне, **щоб збігався набір інструкцій процесора і вистачало ресурсів (оперативної пам'яті, обсягу жорсткого диска та ін.).** Віртуальний сервер можна легко переносити з одного фізичного сервера на інший. До того ж часто це навіть не вимагає зупинки програм.

Віртуалізоване розгортання, яке дозволяє кільком віртуальним машинам працювати на одному фізичному сервері, мало вирішити цю проблему. Як і раніше, на сервері є ОС. На додаток до цього є гіпервізор, який запускає кілька віртуальних машин із власними ОС для розгортання потрібної програми. Це забезпечує більш ефективне використання ресурсів, кращу масштабованість і зниження витрат на обладнання. Але для роботи ОС віртуальної машини потрібні цінні ресурси, а програма не повністю оптимізована.

Контейнери вирішують цю проблему. Вони схожі на віртуальні машини, але мають однакову операційну систему. Вони не покладаються на базову інфраструктуру, тому ними можна керувати та розгортати їх на будь-якому апаратному забезпеченні — у вашій локальній системі, локальній системі, з «голим металом» або в хмарі.

У кожній віртуальній машині запускається ядро операційної системи (наприклад, Linux чи Windows). Для окремої програми запускати своє ядро операційки, з погляду споживаних ресурсів, досить невигідно. Розв'язанням цієї проблеми стала поява механізму використання контейнерів .

Суть контейнера та сама, що й у віртуальної машини: **ми запускаємо програми ізольовано одна від одної**. Але у разі застосування контейнерів **ми не запускаємо ядро операційної системи на кожен із них.** Ізоляція контейнерів забезпечується вбудованими інструментами вже наявної операційної системи.

Розгортання контейнерів є найефективнішим методом, який дає багато переваг, але вам все одно потрібно підтримувати свої контейнери та стежити, щоб не було простоїв. Саме цього Kubernetes може допомогти вам досягти.

Розглянемо детальніше віртуалізацію та контейнерізацію.

Віртуалізація - надання набору обчислювальних ресурсів або їх логічного об'єднання, абстраговані від апаратної реалізації, і забезпечує при цьому логічну ізоляцію один від одного обчислювальних процесів, які виконуються на одному фізичному ресурсі. Прикладом використання віртуалізації є можливість запуску декількох операційних систем на одному комп'ютері: при тому кожен з примірників таких гостьових операційних систем працює зі своїм набором логічних ресурсів (процесорних, оперативної пам'яті, пристроїв зберігання), наданням яких із загального пулу, доступного на рівні обладнання, управляє хостова операційна система - гіпервізор. Якщо говорити про технології, то віртуалізація - це в широкому сенсі відділення уявлення системи від її реалізації. Тобто віртуалізація присутня в будь-якому аспекті ІТ, включаючи операційні системи, сховища даних, веб-браузери та багато іншого.

Існує декілька основних варіантів віртуалізаії, а саме два підходи за допомогою яких ми маємо змогу створювати незалежні простори, які в свою чергу будуть ізольовані один від одного на одному фізичному сервері.

Першим є створення віртуальних машин, для роботи яких потрібен гіпервізор, а другим підходом є створення віртуальних контейнерів. У випадку з віртуальними машинами для кожної використовується власна гостьова ОС, а у випадку з віртуальними контейнерами застосовується ядро хостової ОС. Завдяки цьому, перший підхід дозволяє створювати неоднорідні середовища для обчислень на одному комп'ютері, другий підхід – лише однорідні.

Але оскільки віртуальні машини включають в себе операційну систему, їх розмір буває занадто великим і може досягати кілька гігабайт. Ще одним недоліком віртуальних машин можна вважати те, що для завантаження операційної системи і встановлення програми, яке в ній розміщено, потрібно доволі багато часу в порівнянні з контейнерами. В свою чергу, контейнери більш легкі і, в більшості випадків, їх розмір вимірюється в мегабайтах. В порівнянні продуктивності контейнерів з віртуальними машинами, контейнери можна запускати майже миттєво. Тому при виборі між контейнерами та віртуальними машинами слід враховувати цілі, які намагаєтеся досягти.

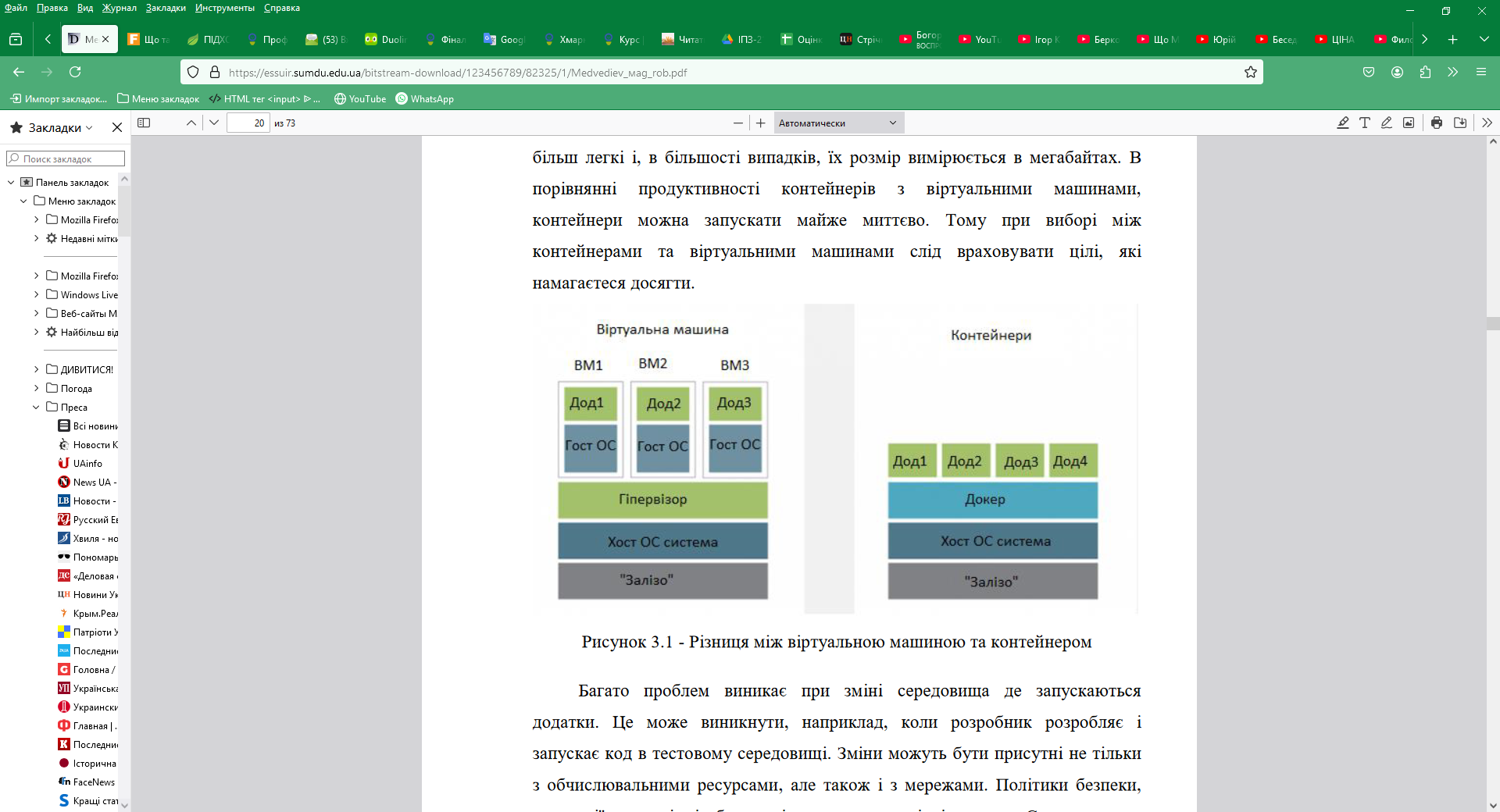


Рисунок 1 - Різниця між віртуальною машиною та контейнером

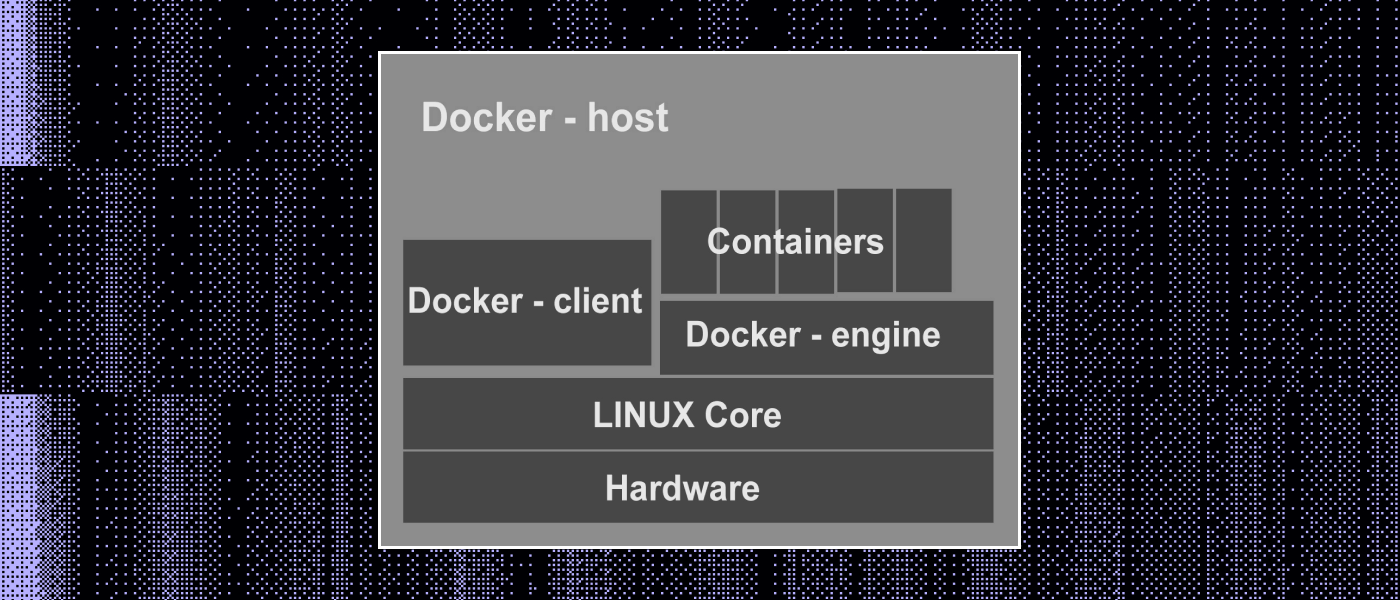
**Управління інфраструктурою, що містить безліч контейнерів, називають**оркестрацією .

**Найпопулярніше рішення для роботи з контейнерами — це [Docker](https://www.docker.com/" \t "blank)**, контейнеризатор застосунків, що дає змогу запакувати їх з усім оточенням та залежностями.

Контейнери в Docker створюються на основі образів, які по суті є шаблонами. **Образ — це набори файлів програми та інсталяційних команд.** Щоб запустити свою програму через докер, формують Dockerfile, у якому є покроковий опис установки на комп'ютер: на основі цієї інформації створюється образ із програмою.

Образ можна встановити на сервер та на його базі запустити контейнер. Він описується одним файлом, що дуже зручно, тому що його можна записати в Git-репозиторій. Файл версіонується, всі зміни в ньому фіксуються, і можна робити збирання нових версій контейнера через систему Continuous Integration (наприклад, GitLab CI).

Docker запускається як системна служба на сервері під керуванням Windows чи Linux. Ця служба відповідає за запуск контейнера, зупинку та інше керування.

У певному сенсі контейнер можна розглядати як «пісочницю» для застосунків, якесь ізольоване середовище, у якому можна запускати застосунки з обмеженими правами доступу до ресурсів системи. У контейнері міститься ізольоване оточення — файлова система, бібліотеки, залежності та навіть операційна система (у випадку контейнера Docker).

При цьому **контейнери залучають загальне ядро ОС-хоста, тому вони більш легкі, якщо порівнювати з віртуальними машинами**, де кожна має власне ядро. Контейнери також дають змогу упаковувати застосунок та його залежності в одну одиницю, що забезпечує цілісність даних та переносність між різними середовищами.

Концепція контейнерів виявилася настільки успішною, що великі компанії, як-от Google, Amazon, Microsoft та Docker, об'єдналися, щоб розробити загальний стандарт формату образів контейнерів. Це спричинило створення проєкту OCI (Open Container Initiative), який визначає стандарти для контейнерів та образів, базуючись на ідеях Docker.

При цьому **контейнери залучають загальне ядро ОС-хоста, тому вони більш легкі, якщо порівнювати з віртуальними машинами**, де кожна має власне ядро. Контейнери також дають змогу упаковувати застосунок та його залежності в одну одиницю, що забезпечує цілісність даних та переносність між різними середовищами.

Концепція контейнерів виявилася настільки успішною, що великі компанії, як-от Google, Amazon, Microsoft та Docker, об'єдналися, щоб розробити загальний стандарт формату образів контейнерів. Це спричинило створення проєкту OCI (Open Container Initiative), який визначає стандарти для контейнерів та образів, базуючись на ідеях Docker.

Переваги контейнерів:

* Гнучке середовище. Основна перевага у використанні контейнерів полягає в їх швидкості створення, на відміну екземплярів віртуальних машин.
* Підвищена продуктивність. Швидкість розробки досягається за рахунок того, що контейнери усувають мережеві залежності й конфлікти. Кожен контейнер може бути незалежно оновлений без будь-яких проблем тому їх можна розглядатися як окремий мікросервіс.
* Версіонування. Управління версіями дозволяє стежити за відмінностями.
* Переносимість. Контейнери інкапсулюють всі необхідні залежності, такі як бібліотеки, необхідні для запуску програми. Це дозволяє переносити контейнера з одного середовища в інше.
* Стандартизація. Контейнери засновані на відкритих стандартах і можуть працювати в основних дистрибутивах Linux, Microsoft.
* Безпека. Контейнери ізолюють процеси один від іншого і від базової операційної системи. Таким чином, будь-яке оновлення не впливає на роботу іншого контейнера.

Недоліки контейнерів:

* Підвищена складність: при великій кількості контейнерів, які працюють з додатком, збільшується складність. Управління безліччю контейнерів досить складне завданням у виробничому середовищі. Такі інструменти, як Kubernetes і Mesos, полегшують управління великою кількістю працюючих контейнерів.
* Також до мінусів можна віднести складність, яка полягає в тому, що зазвичай в контейнер додається більше ресурсів, ніж потрібно – це призводить до розростання образу і великому розмірі контейнера.
* Підтримка Native Linux: основна кількість контейнерних технологій, таких як Docker, основані на Linux-контейнерах. Тому виконання цих контейнерів в середовищі Microsoft – досить складний процес, а їх щоденне використання викликає складності в порівнянні з початковим запуском цих додатків на Linux.
* Незрілість: контейнери - доволі нова технологія на ринку.

Віртуалізація на рівні операційної системи - це спосіб віртуалізації, при якому операційна система допускає роботу декількох ізольованих контейнерів, а не тільки одного. Вони виглядають як справжні комп'ютери з точки зору програм, запущених в них. Підхід корисний, коли необхідно налаштувати декілька операційних систем з ідентичними параметрами. Різні програми можуть бути встановлені і працювати ідентично, як якщо ми запускаємо додаток на операційній системі хоста. Ресурси, призначені контейнеру, доступні лише йому. Для створення контейнерів операційної системи ми можемо використовувати такі контейнерні технології, як VServer, Jails ,LXC, OpenVZ, Solaris , Linux, BSD.

Контейнери додатків: віртуалізація додатків - це програмна технологія, яка інкапсулює комп'ютерні програми з базової операційної системи, на якій вона виконана. Повністю віртуалізований додаток не встановлюється в традиційному сенсі, хоча він все одно виконується так, як якщо б він був встановлений. Додаток поводиться під час виконання, так як ніби він безпосередньо взаємодіє з вихідної операційною системою і всіма ресурсами, якими він управляє, але може бути ізольованим в різному ступені. Контейнери додатків призначені для упаковки і запуску служб як одного процесу, тоді як в контейнерах ОС можуть виконуватися кілька сервісів і процесів. Контейнерні технології, такі як Docker і Rocket, є прикладами контейнерів для додатків.

***Оркестрація контейнерів***

Контейнерні платформи, такі як Docker, зараз є дуже популярними для упаковки додатків, які базуються на мікросервісній архітектурі. Оркестрації - це координація взаємодії декількох контейнерів. Звичайно, можна працювати і без оркестрації - ніхто не забороняє створити контейнер, в якому будуть запущені всі необхідні процеси. Однак в цьому випадку ви будете позбавлені гнучкості, масштабованості, а також виникнуть питання безпеки, оскільки запущені в одному контейнері процеси не будуть ізольовані і зможуть впливати друг на друга. Оркестрація дозволяє створювати інформаційні системи з безлічі контейнерів, кожен з яких відповідає тільки за одну певну задачу, а спілкування здійснюється через мережеві порти і загальні каталоги. При необхідності кожен такий контейнер можна замінити іншим, що дозволяє, наприклад, швидко перейти на іншу версію бази даних при необхідності.

Існують різні платформи для оркестрації контейнерів. Вони дозволяють реалізувати зручні та ефективні засоби розгортання контейнерних систем, побудови єдиної централізованої консолі для застосування політик управління.

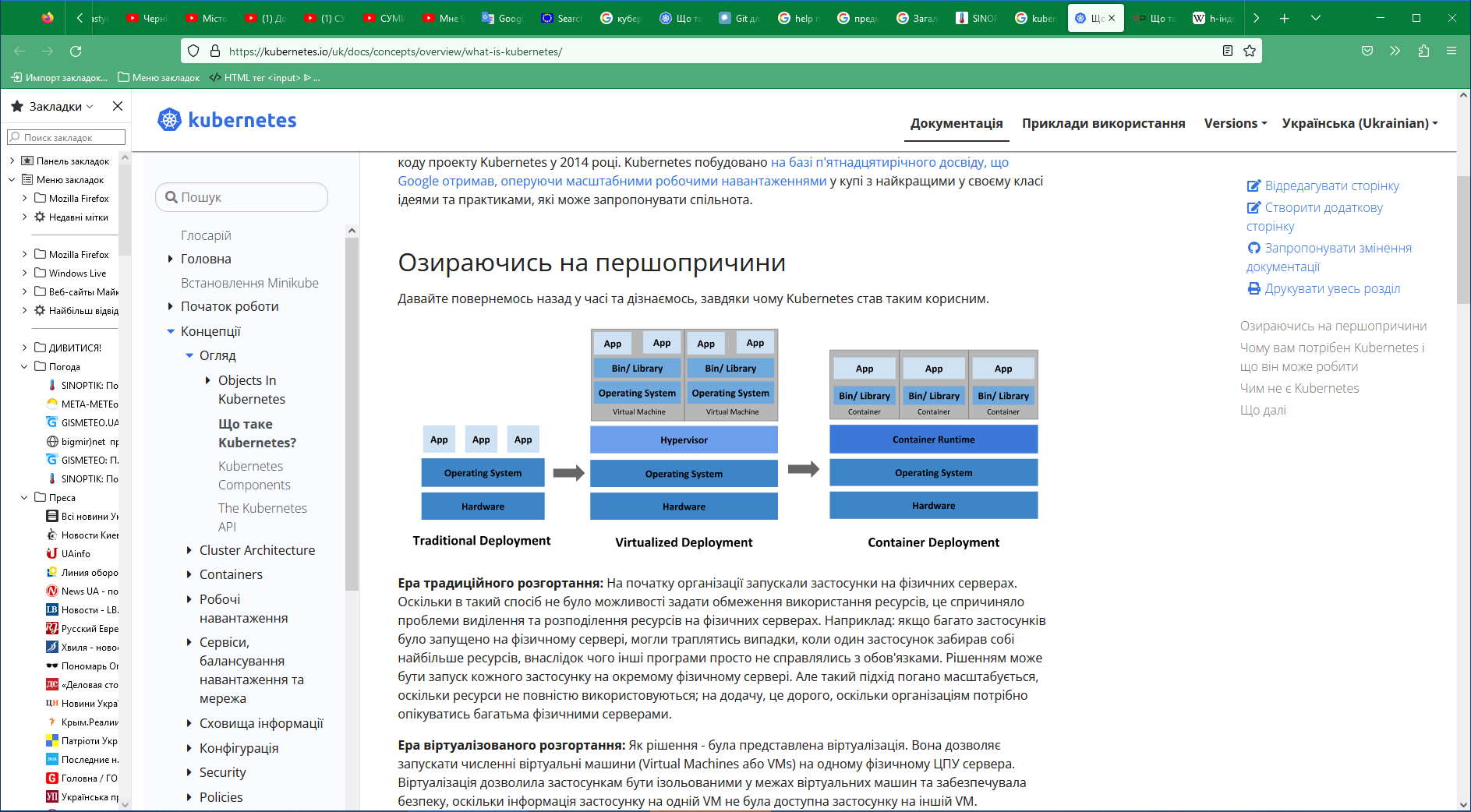
Найбільш популярним є Kubernetes - платформа з відкритим вихідним кодом, спочатку розроблена Google і в даний час підтримувана Cloud Native Computing Foundation. Kubernetes підтримує як декларативну конфігурацію, так і автоматизацію. Це може допомогти автоматизувати розгортання, масштабування і управління контейнерної робочим навантаженням і послугами. API Kubernetes допомагає встановити зв'язок між користувачами, компонентами кластеру і зовнішніми компонентами сторонніх виробників.

Рівень управління Kubernetes і самі вузли виконуються на групі вузлів, які разом утворюють кластер. Робоче навантаження додатка складається з одного або декількох модулів, які виконуються на вузлі (вузлах) Worker.

Рівень управління контролює групи контейнерів (Pod-и) і робочі вузли. До особливостей можна віднести :

* служби виявлення і балансування навантаження;
* оркестрації системи зберігання даних;
* автоматизовані розгортання і відкати;
* горизонтальне масштабування;
* управління безпекою і конфігурацією;
* самовідновлення;
* пакетне виконання;
* подвійний стек IPv4 / IPv6.

**Kubernetes**  це платформа з відкритим вихідним кодом для управління контейнеризованими робочими навантаженнями та супутніми службами. Її основні характеристики - кросплатформенність, розширюваність, успішне використання декларативної конфігурації та автоматизації. Вона має гігантську, швидко прогресуючу екосистему.



**Ера традиційного розгортання:** На початку організації запускали застосунки на фізичних серверах. Оскільки в такий спосіб не було можливості задати обмеження використання ресурсів, це спричиняло проблеми виділення та розподілення ресурсів на фізичних серверах. Наприклад: якщо багато застосунків було запущено на фізичному сервері, могли траплятись випадки, коли один застосунок забирав собі найбільше ресурсів, внаслідок чого інші програми просто не справлялись з обов'язками. Рішенням може бути запуск кожного застосунку на окремому фізичному сервері. Але такий підхід погано масштабується, оскільки ресурси не повністю використовуються; на додачу, це дорого, оскільки організаціям потрібно опікуватись багатьма фізичними серверами.

**Ера віртуалізованого розгортання:** Як рішення - була представлена віртуалізація. Вона дозволяє запускати численні віртуальні машини (Virtual Machines або VMs) на одному фізичному ЦПУ сервера. Віртуалізація дозволила застосункам бути ізольованими у межах віртуальних машин та забезпечувала безпеку, оскільки інформація застосунку на одній VM не була доступна застосунку на іншій VM.

Віртуалізація забезпечує краще використання ресурсів на фізичному сервері та кращу масштабованість, оскільки дозволяє легко додавати та оновлювати застосунки, зменшує витрати на фізичне обладнання тощо. З віртуалізацією ви можете представити ресурси у вигляді одноразових віртуальних машин.

Кожна VM є повноцінною машиною з усіма компонентами, включно з власною операційною системою, що запущені поверх віртуалізованого апаратного забезпечення.

**Ера розгортання контейнерів:** Контейнери схожі на VM, але мають спрощений варіант ізоляції і використовують спільну операційну систему для усіх застосунків. Саме тому контейнери вважаються "легкими", в порівнянні з віртуалками. Подібно до VM, контейнер має власну файлову систему, ЦПУ, пам'ять, простір процесів тощо. Оскільки контейнери вивільнені від підпорядкованої інфраструктури, їх можна легко переміщати між хмарними провайдерами чи дистрибутивами операційних систем.

Контейнери стали популярними, бо надавали додаткові переваги, такі як:

* Створення та розгортання застосунків за методологією Agile: спрощене та більш ефективне створення образів контейнерів у порівнянні до використання образів віртуальних машин.
* Безперервна розробка, інтеграція та розгортання: забезпечення надійних та безперервних збирань образів контейнерів, їх швидке розгортання та легкі відкатування (за рахунок незмінності образів).
* Розподіл відповідальності команд розробки та експлуатації: створення образів контейнерів застосунків під час збирання/релізу на противагу часу розгортання, і як наслідок, вивільнення застосунків із інфраструктури.
* Спостереження не лише за інформацією та метриками на рівні операційної системи, але й за станом застосунку та іншими сигналами.
* Однорідність середовища для розробки, тестування та робочого навантаження: запускається так само як на робочому комп'ютері, так і у хмарного провайдера.
* ОС та хмарна кросплатформність: запускається на Ubuntu, RHEL, CoreOS, у власному дата-центрі, у Google Kubernetes Engine і взагалі будь-де.
* Керування орієнтоване на застосунки: підвищення рівня абстракції від запуску операційної системи у віртуальному апаратному забезпеченні до запуску застосунку в операційній системі, використовуючи логічні ресурси.
* Нещільно зв'язані, розподілені, еластичні, вивільнені мікросервіси: застосунки розбиваються на менші, незалежні частини для динамічного розгортання та управління, на відміну від монолітної архітектури, що працює на одній великій виділеній машині.
* Ізоляція ресурсів: передбачувана продуктивність застосунку.
* Використання ресурсів: висока ефективність та щільність.

Kubernetes надає каркас для еластичного запуску розподілених систем. Він опікується масштабуванням та аварійним відновленням вашого застосунку, пропонує шаблони розгортань тощо. Наприклад, Kubernetes дозволяє легко створювати розгортання за стратегією canary (розгортання нових версій застосунку поряд зі стабільними, робочими версіями.) у системі.

Kubernetes надає:

* **Виявлення сервісів та балансування навантаження** Kubernetes може надавати доступ до контейнера, використовуючи DNS-ім'я або його власну IP-адресу. Якщо контейнер зазнає завеликого мережевого навантаження, Kubernetes здатний збалансувати та розподілити його таким чином, щоб якість обслуговування залишалась стабільною.
* **Оркестрація сховища інформації** Kubernetes дозволяє вам автоматично монтувати системи збереження інформації на ваш вибір: локальні сховища, рішення від хмарних провайдерів тощо.
* **Автоматичне розгортання та відкатування** За допомогою Kubernetes ви можете описати бажаний стан контейнерів, що розгортаються, і він регульовано простежить за виконанням цього стану. Наприклад, ви можете автоматизувати в Kubernetes процеси створення нових контейнерів для розгортання, видалення існуючих контейнерів і передачу їхніх ресурсів на новостворені контейнери.
* **Автоматичне розміщення задач** Ви надаєте Kubernetes кластер для запуску контейнеризованих задач і вказуєте, скільки ресурсів ЦПУ та пам'яті (RAM) необхідно для роботи кожного контейнера. Kubernetes розподіляє контейнери по вузлах кластера для максимально ефективного використання ресурсів.
* **Самозцілення** Kubernetes перезапускає контейнери, що відмовили; заміняє контейнери; зупиняє роботу контейнерів, що не відповідають на задану користувачем перевірку стану, і не повідомляє про них клієнтам, допоки ці контейнери не будуть у стані робочої готовності.
* **Управління безпекою (секретами) та конфігурацією** Kubernetes дозволяє вам зберігати та керувати чутливою інформацією, такою як паролі, OAuth токени та SSH ключі. Ви можете розгортати та оновлювати секрети та конфігурацію без перезбирання образів ваших контейнерів, не розкриваючи секрети в конфігурацію стека.

*Література*

1. Що таке Kubernetes? | Короткий вступ простою мовою. URL: <https://nt.ua/blog/what-is-kubernetes> (дата звернення 11.07.2024)

# Чому не варто використовувати Kubernetes. URL: https://dou.ua/forums/topic/44569/(дата звернення 11.07.2024)

# Серверний диригент: що таке Kubernetes і як він працює. URL: https://robotdreams.cc/uk/blog/397-serverniy-dirigent-shcho-take-kubernetes-i-yak-vin-pracyuye (дата звернення 11.07.2024)

# Документація Kubernetes. URL: https://kubernetes.io/uk/docs/home/ (дата звернення 11.07.2024)

# Підручник Kubernetes для початківців: основи, функції, Archiтектура. URL: <https://www.guru99.com/uk/kubernetes-tutorial.html> (дата звернення 11.07.2024)