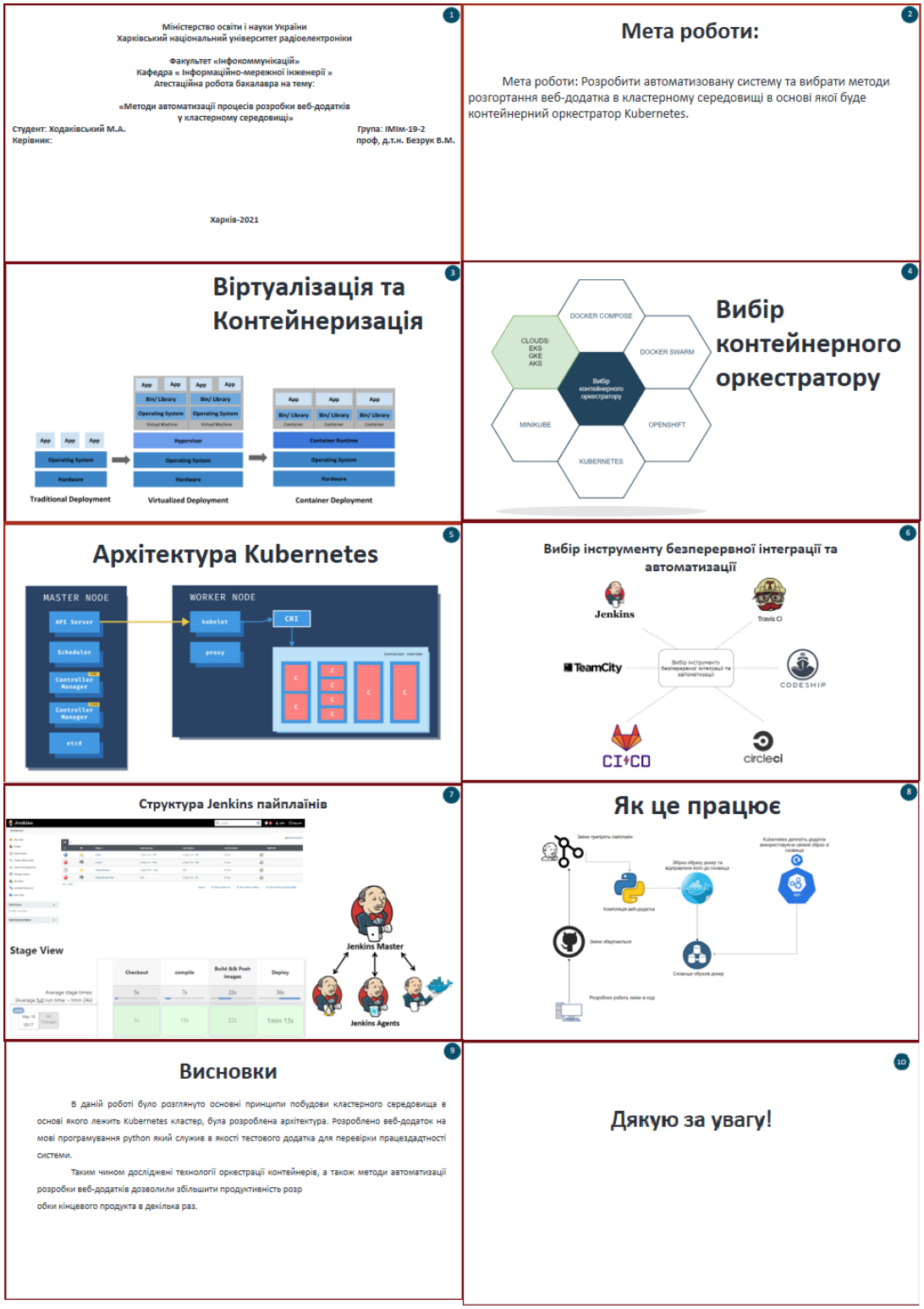
ДОДАТОК А СЛАЙДИ ПРЕЗЕНТАЦІЇ



ДОДАТОК Б ПУБЛІКАЦІЯ ПО ТЕМІ РОБОТИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

МАТЕРІАЛИ
XXIV МІЖНАРОДНОГО МОЛОДІЖНОГО ФОРУМУ

**«РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ
У XXI СТОЛІТТІ»**

7 – 9 квітня 2020 р.

Том 4

КОНФЕРЕНЦІЯ
«ПЕРСПЕКТИВИ РОЗВИТКУ ІНФОКОМУНІКАЦІЙ ТА ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНИХ ТЕХНОЛОГІЙ»

Харків 2020

ВПРОВАДЖЕННЯ ХМАРНОЇ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ НА ПРИКЛАДІ КОНТЕЙНОРНОЇ ПРОГРАМИ OPENSHIFT

Ходаківський М.М.

Науковий керівник: - к.т.н., доц. Бондарь Д. В., ст. викл. – Малінін О. П. Харківський національний університет радіоелектроніки (61166, Харків, просп. Науки, 14, каф. інформаційно-мережної інженерії, (057) 702-14-29) e-mail: mykola.khodakivskyi@nure.ua, тел. 098-88-61-265

Today there is a problem with speed, complexity and efficiency of development of the final software product, for many years the developers have written large web applications with the so-called monolithic method, this is when developing a large application that stores all the modules and pieces of code, it was and is quite convenient in writing, but this method has significant disadvantages, first of all, if an application fails one module or another, the whole application ceases to function, which is a critical aspect when looking from a business standpoint, a difficult process debugging and application updates.

Microservice architecture is the architecture of the current lion's fate of all the most popular resources, or projects that use this particular application building system. The principle of microservice architecture is that the whole application is broken down into services, for example, the application is authorized, in the service architecture it can be made a separate module, etc. The advantages of this principle are the stability of the application, if one service fails it will be easy to repair, easy to update the version of the application, also a disadvantage is the difficult process of debugging and updating the application.

На сьогоднішній день існує проблема зі швидкістю, складністю та ефективністю розробки кінцевого програмного продукта, багато років розробники писали великі веб-додатки так названим монолітним методом, це коли розробляється великий додаток який в собі зберігає всі модулі та частинки коду, це було і є досить зручно в написанні, але такий метод зберігає значні недоліки, перш за все якщо в додатку вийде з ладу тий чи інший модуль працездатність всього додатка стане під загрозою, що є критичним аспектом якщо дивитись з позиції бізнесу, також як недолік варто віднести досить важкий процес відлатки та оновлення додатку.

Мікросервісна архітектура – це архітектура сьогодення львина доля всіх найпопулярнійших ресурсів, або проектів використовує саме цю систему побудову додатків. Принцип мікросервісної архітектури в тому, що весь додаток розбивається на сервіси, наприклад додаток має авторизацію, в сервісній архітектурі його можливо зробити окремим сервісом. Переваги такого принципу є стабільність додатку, якщо один сервіс вийде з ладу його буде легко полагодити, також легко оновлювати версію додатку. Як недолік можна віднести досить важке налаштування і в цілому міжсервісна взаємодія.

Openshift потрібен для створення мікросервісної архітектури, в основі кожного сервіса лежить контейнер – це ізольована операційна система в середні якої є певна програма, тобто це і є сервіс. На рис 1. Зображена основна концепція побудови мікросервісної архітектури на основі openshift. Основним елементом системи є "MASTER" так звана система яка контролює всі процеси що проходять в нашій системі, базовим елементом виступає "Node" – це фізичний комп'ютер в середині якої є "Pod" – це область в якому можуть зберігатися наші ізольовані контейнери з нашими сервісами, за допомогою openshift ми маємо змогу пов'язувати певні контейнери один з одним, таким чином створювати бізнес логіку та міжсервісну архітектуру також за допомогою openshift можливо зручно дублювати наші сервіси для більш відмовостійкості.

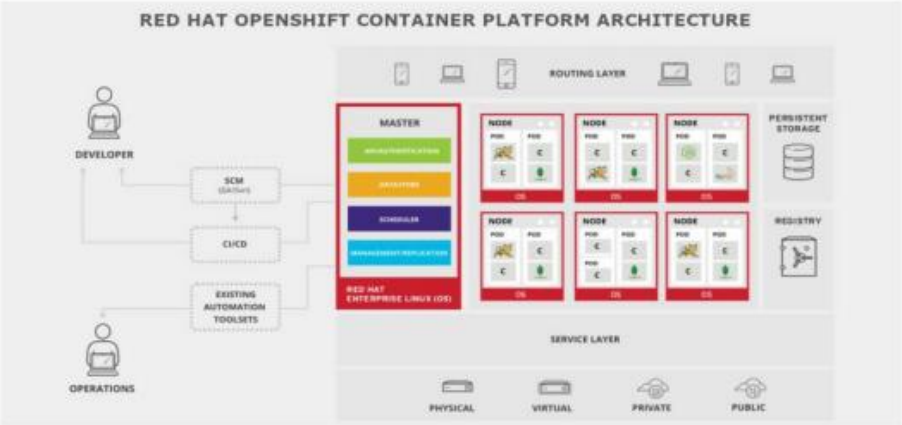


Рисунок1. Концепція мікросервісної архітектури Openshift

Процес роботи буде виглядати таким чином, розробник працює над певним сервісом, він робить так званий "patch-set" тобто зміну в своєму коді, код попадає до блоку ci\cd що на малюнку там він компілюється, тестується, і на виході він попадає в блок "Regestry" це є кінцевий результат "артефакт" тобто ми маємо snap-shot версію сервіса, далі в ручному, або в автоматичному режимі потрібно оновити окремий сервіс вказавши йому в налаштуваннях шлях до "артефакта" таким чином можна дуже зручно та безпечно поступово оновлювати та розробляти додаток, це дуже зручно так як певна група розробників займаються тільки окремим сервісом.

Перелік джерел

1. Книга "Openshiftinaction" [Електроний ресурс]: https://www.manning.com/books/openshift-in-action

2. Ознайомлення з Openshift [Електроний ресурс]: https://habr.com/ru/ article/454624/

ДОДАТОК В ПУБЛІКАЦІЯ ПО ТЕМІ РОБОТИ

ВПРОВАДЖЕННЯ ХМАРНОЇ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ НА ПРИКЛАДІ ОРКЕСТРАТОРА KUBERNETES

Ходаківський М.А.
Науковий керівник: - д.т.н., проф. Безрук В. М., ст. викл. – Малінін О.
П. Харківський національний університет радіоелектроніки (61166, Харків, просп. Науки, 14, каф. інформаційно-мережної інженерії, (057) 702-14-29)
e-mail: mykola.khodakivskyi@nure.ua, тел. 098-88-61-265

Today there is a problem with speed, complexity and efficiency of development of the final software product, for many years the developers have written large web applications with the so-called monolithic method, this is when developing a large application that stores all the modules and pieces of code, it was and is quite convenient in writing, but this method has significant disadvantages, first of all, if an application fails one module or another, the whole application ceases to function, which is a critical aspect when looking from a business standpoint, a difficult process debugging and application updates.я

Microservice architecture is the architecture of the current lion's fate of all the most popular resources, or projects that use this particular application building system. The principle of microservice architecture is that the whole application is broken down into services, for example, the application is authorized, in the service architecture it can be made a separate module, etc. The advantages of this principle are the stability of the application, if one service fails it will be easy to repair, easy to update the version of the application, also a disadvantage is the difficult process of debugging and updating the application.

На сьогоднішній день існує проблема зі швидкістю, складністю та ефективністю розробки кінцевого програмного продукта, багато років розробники писали великі веб-додатки так названим монолітним методом, це коли розробляється великий додаток який в собі зберігає всі модулі та частинки коду, це було і є досить зручно в написанні, але такий метод має певні недоліки, перш за все якщо в додатку вийде з ладу тий чи інший модуль тоді працездатність всього додатка стане під загрозою, що є критичним аспектом, якщо дивитись з позиції бізнесу то як недолік варто віднести досить важкий процес відлатки та оновлення додатку.

Мікросервісна архітектура – це архітекура сьогодення львина доля всіх найпопулярнійших ресурсів, або проектів використовує саме цю систему методику побудови додатків. Принцип мікросервісної архітектури в тому, що весь додаток розподіляється на мікросервіси, наприклад додаток має авторизацію, в сервісній архітектурі його можливо зробити окремим мікросервісом. Переваги такого принципу є стабільність додатку, якщо один мікросервісом вийде з ладу його буде легко полагодити, також легко

оновлювати версію додатку. Як недолік можна віднести досить важке налаштування і в цілому міжсервісна взаємодія.

Kubernetes – потрібен для управління мікросервісною архітектурою,. На рис 1. Зображена основна концепція побудови мікросервісної архітектури на основі kubernetes. Основним елементом системи є “Master-Node” окрема фізична машина, або віртуальна машина, яка контролює всі процеси що проходять в нашій системі, базовим елементом виступає “Node” – це фізична машина, або віртуальная машина, на якій може виконуватися робота так само як і на “Master-Node”. В середені “Node” є “Pod” – це область в якому можуть зберігатися наші ізольовані контейнери з нашими мікросервісами, за допомогою kubernetes ми створюєму логічну мережну архітектуру в середені “Node” за допомогою сущності “Service”, таким чином створювати бізнес логіку та міжсервісну архітектуру також за допомогою kubernetes можливо зручно дублювати наші сервіси для більш відмовостійкості.

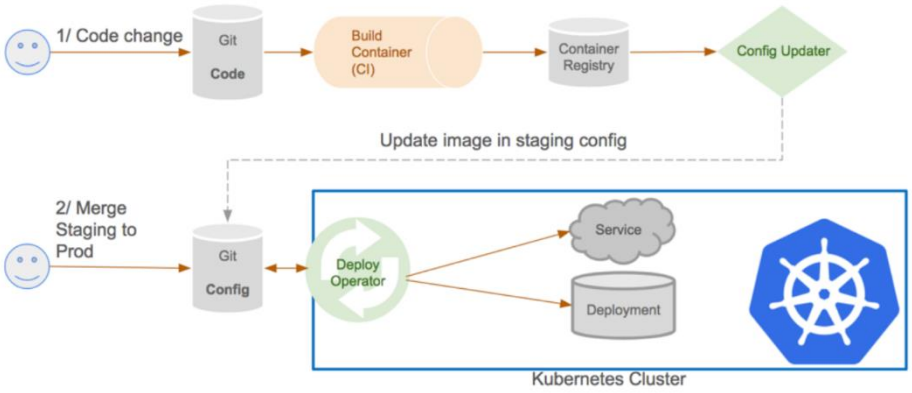


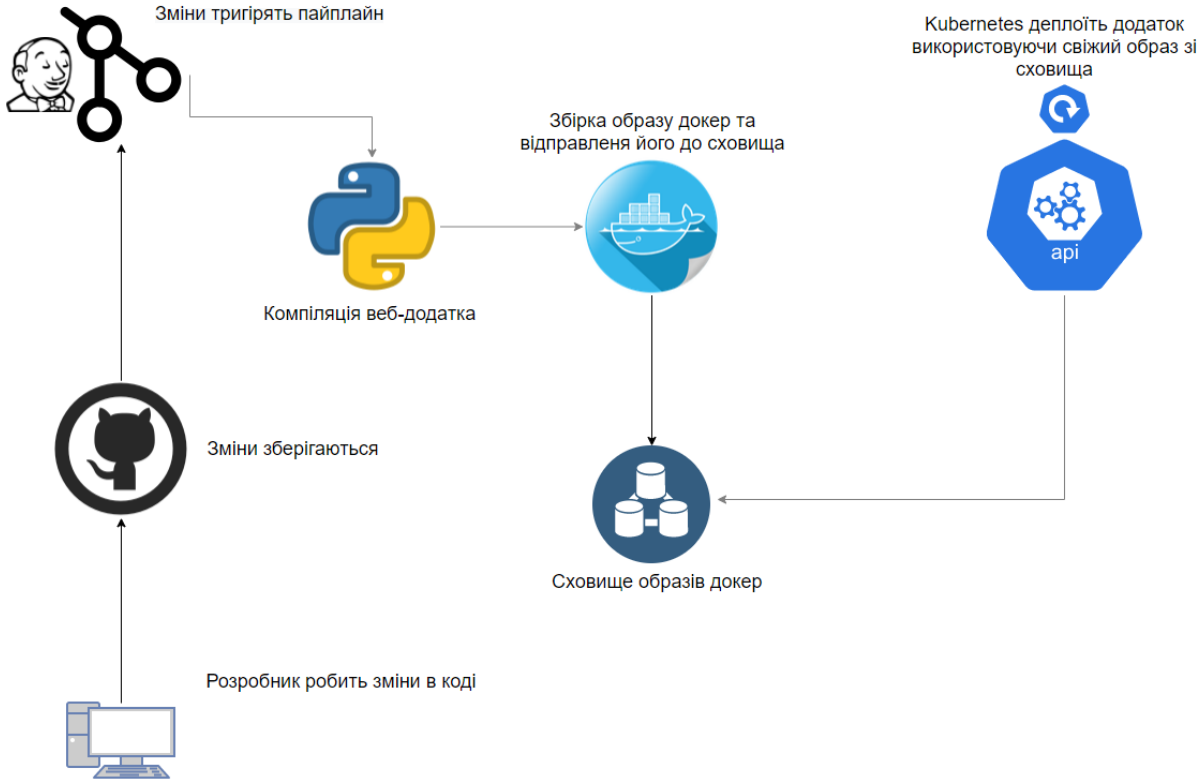
Рисунок1. Концепція мікросервісної архітектури Kubernetes

Процес роботи буде виглядати таким чином, розробник працює над певним мікросервісом, він робить так званий “patch-set” тобто зміну в своєму коді, код попадає до блоку “ci” що на малюнку там він компілюється, тестується, і на виході він попадає в блок “container-regestry” , далі в ручному, або в автоматичному режимі потрібно оновити окремий мікросервіс вказавши йому в налаштуваннях шлях до “артефакта” таким чином можна дуже зручно та безпечно поступово оновлювати та розробляти додаток.

Перелік джерел

1. Ознайомлення з Kubernetes [Електроний ресурс]: https://kubernetes.io/ru/docs/concepts/overview/what-is-kubernetes/
2. Основи Kubernetes [Електроний ресурс]: https://habr.com/ru/post/258443/

ДОДАТОК Г СХЕМА ІНТЕГРАЦІЇ CICD ПРОЦЕСУ



Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	86 с.
	Графічні документи	
2.	Слайди презентації	10

Змін.	Арк.	Номер докум.	Підпис	Дата				
Розроб.		Ходаківський			Методи автоматизації процесів розробки веб-додатків у кластерному середовищі Відомість кваліфікаційної роботи	Літ.	Аркуш	Аркушів
Перевір.		Безрук				У	1	1
						ХНУРЕ кафедра ІМІ		
Н. контр.		Безрук						
Затв.		Безрук						