UDC 003.26 **doi: 10.20998/2522-9052.2020.2.21**

V. Pevnev, Yu. Trehub

National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine

ANALYSIS AND RESEARCH OF WELL-KNOWN ORCHESTRATION SYSTEMS FOR THE CONSTRUCTION OF MICROSERVICE INFRASTRUCTURE

Abstract. Modern orchestration systems are being studied. Using different virtualization methods is it a good way to reduce construction time, equipment costs and support system operations during infrastructure construct. The article provides a detailed analysis of existing virtualization tools. A comparative analysis of modern solutions shows that the Kubernetes tool is one of the best tools for orchestration. The features of sharing orchestration systems together with different Cloud providers are presented in the article, comparisons of the cloud providers themselves are presented too.

Keywords: virtualization technologies; orchestration systems; Docker; Docker Swarm; Kubernetes; Amazon Web Services, Google Cloud Platform, Microsoft Azure.

Introduction

Due to the development of information technology, the amount of data processed is increasing every day. That is why the problems of choosing a reliable system for storing and processing user data are also increasing.

Based on the requirements for guarantee systems, we can conclude that it is necessary to ensure the availability and integrity of the information that circulates in different information systems. Methods for ensuring the integrity of information are given in sufficient detail in [1].

Today we have many infrastructure systems, but one of the most popular is Cloud systems. The choice of the system depends on the requirements that a particular organization applies to own IT infrastructure. A comparative analysis of the market share of leading cloud providers shows that 35% of the market is owned by AWS-based systems, 11% by Microsoft Azure and 7% by Google Cloud Services. It was noted that growth was an average of about 2% over 2018, with only Microsoft Azure rising to 3% [2]. All of these providers allow users to create reliable and easily scalable infrastructure.

However, any infrastructure requires a large amount of resources. That is why it is necessary to use solutions that will help to use these resources efficiently, without losing all the advantages provided by Cloud provider. The virtualization helps users with creating a secure, reliable and scalable infrastructure.

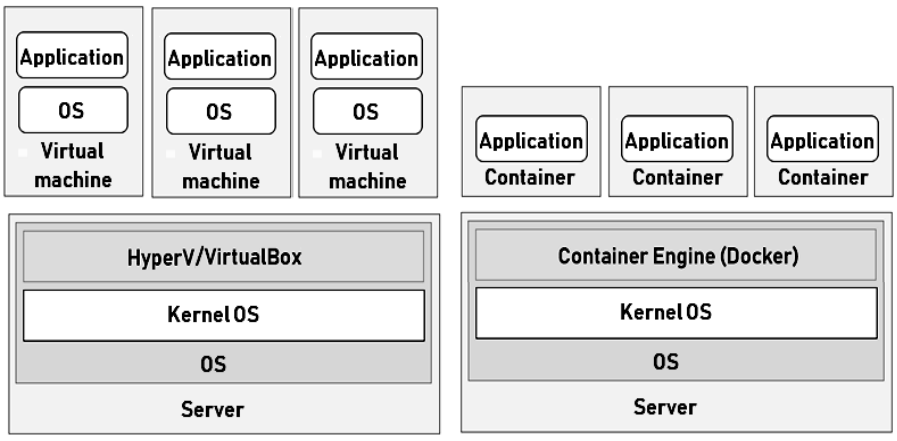
The main goal of the article is the comparable analysis of well-known orchestration systems and choosing the most secure and reliable system for the construction of microservice infrastructure.

Analysis of orchestration systems

The main virtualization technologies are hypervisor-based virtualization (eg, Hyper-V, Virtual Box, VMware) and container virtualization (eg, Docker) (Fig. 1).

Hypervisors are hardware-level virtualization. There is a layer between the host and guest OSs that emulates hardware.

The method of hypervisor virtualization have some disadvantages – the decrease in efficiency and idle computing resources, the complexity of porting various applications between virtual operating systems, the ability to lose all virtual machines in the event of failure of one hypervisor on which they are installed, etc. [2].



a b

Fig. 1. Virtualization technologies (a – virtualization using hypervisor, b – container virtualization)

Container virtualization is a virtualization at the operating system level, not hardware level, when each guest OS uses the same kernel (and in some cases, other parts) as the host OS. This gives the containers a big advantage: they are smaller and more compact in hypervisional guest environments. Container

virtualization uses the capabilities of the kernel to create an isolated environment for processes. Unlike hypervisor-based virtualization, containers do not receive their own virtual hardware, but use host operating system hardware [3]. One of the features of containers is that they can be managed (orchestrated) as a cluster of multiple applications. A system that is a container cluster is called a container orchestration system. The most popular container management systems (or container orchestration systems) are: Kubernetes and OpenShift. There are also other container orchestration systems, such as: Amazon Web Services (AWS) Elastic Compute Cloud (EC2) Container Service (ECS) and others.

Docker is a software platform for quickly developing, testing, and deploying applications in containers. Each container includes everything that you need for the program: libraries, system tools, code, and runtime. Using Docker allows you to use applications faster and more efficiently, standardizes the operations performed by the applications and optimizes the use of resources. With Docker, users get an object that can be run on any platform with high reliability. But part of the professional systems engineers is to avoid using Docker in production because it is not stable tool [4]. Today, Docker is not a reliable tool for using in production, but it continues to be used by many customers.

Docker Swarm is a tool for clustering and managing Docker containers [5]. Clustering is an important function of container technology because it creates a common group of nodes that can be used to achieve a reliability and resiliency, when one of the nodes fails. Containers and nodes are controlled through a manager node, which organizes and plans containers at other worker nodes [6]. Docker Swarm is tightly works with Docker, and developers can easily customize it without installing additional software services.

Kubernetes is an open source platform designed to deploy, scale and manage a Docker or rkt container cluster as a single system across multiple hosts. The project was started by Google in 2014, it has invested one and a half decades of experience with containers [7]. Now Kubernetes has an active community and is supported by companies such as Microsoft, RedHat, IBM and Docker. Kubernetes has many advantages over other container orchestration systems. It speeds up the development process, simplifies and automates deployments and sequential updates [8]. It also manages applications and services, allowing to build a resilient infrastructure that works without downtime. Kubernetes allows to recover lost containers (in the event of a container failure) or to find and restart existing containers on another node (in case of a node loss).

Kubernetes clusters can run on EC2 and integrate with other Amazon services, such as Amazon Elastic Block Storage, Elastic Load Load Balancing, Auto Scaling, etc. Kubernetes is still popular because has flexible architecture, open source innovations [9]. Kubernetes has become one of the most popular orchestration systems, ranked first on the market and used by many real projects with great success [8], [9].

Comparative analysis of the Docker Swarm and Kubernetes orchestration systems

Comparative analysis of Kubernetes and Docker Swarm shows that each container orchestrator has own advantages and disadvantages. If you need a quick installation and simple configuration requirements, Docker Swarm can be a good solution. If the application is complex and uses hundreds of thousands of containers in production, then Kubernetes should be selected. Although both orchestrators have the same functions, but they have a fundamental differences between their work. Below are the biggest differences between these two platforms.

Cluster installation and configuration. Docker Swarm is much easier to set up, it doesn't take a lot if time to deploy a container group. Along with ease of use, Swarm also provides flexibility by allowing any new node to join existing cluster as manager or employee, and move or manage the nodes and their roles. Each node encrypts the data before sharing with neighboring nodes. Shared data encryption keys are stored in the master nodes inside Swarm, only hosts in that cluster can access cluster management.

Kubernetes requires several manual configurations to connect components. Before starting the Kubernetes should also know a lot of the cluster configuration, as the IP addresses of the nodes, which role each node must take, and how many nodes they have in common. Kubernetes allows you manually manage settings and access to containers from the outside. Also, there is a Pod Network Communications Policy the main load is on pods, that contained of one or more containers that are deployed together. Kubernetes assigns each pod an IP address that can be routed from all other pod, even through major servers. Kubernetes network policies set access permissions for pod groups, similar to security groups, which are used to control access to VM instances.

Customize the container. The Swarm API provides much of the functionality from Docker, but does not cover all of commands. It supports many tools that work with Docker, but if the Docker API lacks a specific operation, there is no easy way to get around it with Swarm. Kubernetes uses own client definitions, APIs and YAML files, which are different from the standard docker equivalents. Using them, you can easily manage containers and the Kubernetes cluster.

Scalability. Docker Swarm deploys containers faster than Kubernetes in very large clusters and high clustering stages, allowing you to quickly respond to scaling as needed. You can run new replicas with one upgrade command. The complexity of Kubernetes stems from providing a unified API suite and reliable cluster state guarantees that slow container deployment and scaling.

Accessibility. Kubernetes and Docker Swarm provide high availability of replication services. The same service is deployed to multiple nodes to provide redundancy. If the host that runs the service has shut down, the service is being redistributed on the other

hosts. It makes services are self-renewing. Although any of the orchestral containers can run on the same server, true redundancy requires additional nodes.

Load balancing. Docker Swarm provides built-in load balancing. All containers in a single cluster are connected to a shared virtual network that allows connection from any node to any container. Docker Swarm also does not allow managing the internal balancer on a node and is bound to ports on which containers are deployed. Kubernetes allows you to manage and adjust load balancing. Each service is accessible through a set of rules that can be accessed without worrying about ports, IPs, and other physical settings.

Container updates and cluster restore. The Docker Swarm container is upgraded by telling the planner to use the new image instead. You can then roll out the updates, preventing the service from shutting down and letting you roll back if something goes wrong. Kubernetes processes the upgrade process by gradually tracking the health of the service to maintain accessibility throughout the upgrade process, making changes to one each container incrementally.

Table 1 gives the results of comparative analysis.

The choice of orchestration system depends on the tasks and goals that you want to achive. If you need a simple system that does not require additional configuration and allows you to quickly build a cluster, then you can choose Docker Swarm. But if you want to run more containers in a secure and well-tuned environment, then you need to choose the Kubernetes orchestration system.

Table 1 - **Comparative analysis of Docker Swarm and Kubernetes**

Criterion for comparison	Docker Swarm	Kubernetes
Cluster installation	Easy and quick to install and configure.	It takes some time to install and configure.
Customize the container	Functionality is provided and limited by the Docker API.	Client definitions, APIs and YAML are unique to Kubernetes.
Scalability	Quick deployment and scaling in not very large clusters. Adding a new node does not require additional master node settings.	Quick deploy and scale in very large clusters. Adding a new node requires additional master node settings.
Accessibility	High availability is ensured through the replication of containers, their constant monitoring and maintenance.	
Load balancing	Automated internal load balancing across any cluster node.	You can manually adjust the internal load balancing.
Container updates and cluster restore	Early planning of service maintenance processes during the upgrade.	Progressive updates and health monitoring through updates.

Research on Kubernetes cluster architecture

The main function of Kubernetes is the orchestration of containers, the scheduling of containers of varying capacity on physical and virtual devices. Containers must be packaged efficiently, and they must comply the constraints that imposed by the deployment environment and cluster configuration. In addition, the Kubernetes platform must monitor all running containers and replace those that have failed, failed, or experienced any other difficulties.

The Kubernetes architecture is designed to extend the cluster and consists of two nodes: a master server and a large number of working servers which called worker nodes (Fig. 2). The master node performs the main function and manages the cluster. Worker nodes cannot function properly without a working master node and are only used to deploy Kubernetes objects.

A. The main elements of the Kubernetes cluster

The Kubernetes cluster include:

– node is a cluster server or virtual machine;

– pods are containers that are grouped together, typically, pod includes from 1 to 5 containers;

– services is an abstraction over pods that defines the access policy for them. Service automatically distributes load between nodes, finds and route traffic to the required container;

– etcd is an instance that stores the status of the master server. It provides reliable storage of configuration data and notification of different changes;

– Kubernetes API Server is an API that enables API server to work. It is intended to be a CRUD server with embedded business logic implemented in separate components or in plugins. It basically handles REST operations, checking them and updating the objects in etcd;

– scheduler binds pod to node via call API;

– Kubernetes Controller Manager Server responsible for other cluster level functions. For example, the nodes are detected, managed, and controlled by node controller. As a result, this entity can be divided into separate components to make them independently connected;

– replication controller is a mechanism based on the pod API. Responsible for the number of pods;

– kubectl is a command line interface for managing Kubernetes;

– kubelet manages pod, their containers, images, partitions, etc;

– kube-proxy is a simple proxy balancer. This service is started on every note and configured in the Kubernetes API. Kube-Proxy can perform the simplest redirection;

– volumes is a directory (section) with data in it that is available in the container. This directory is the shared between the container and the nod.

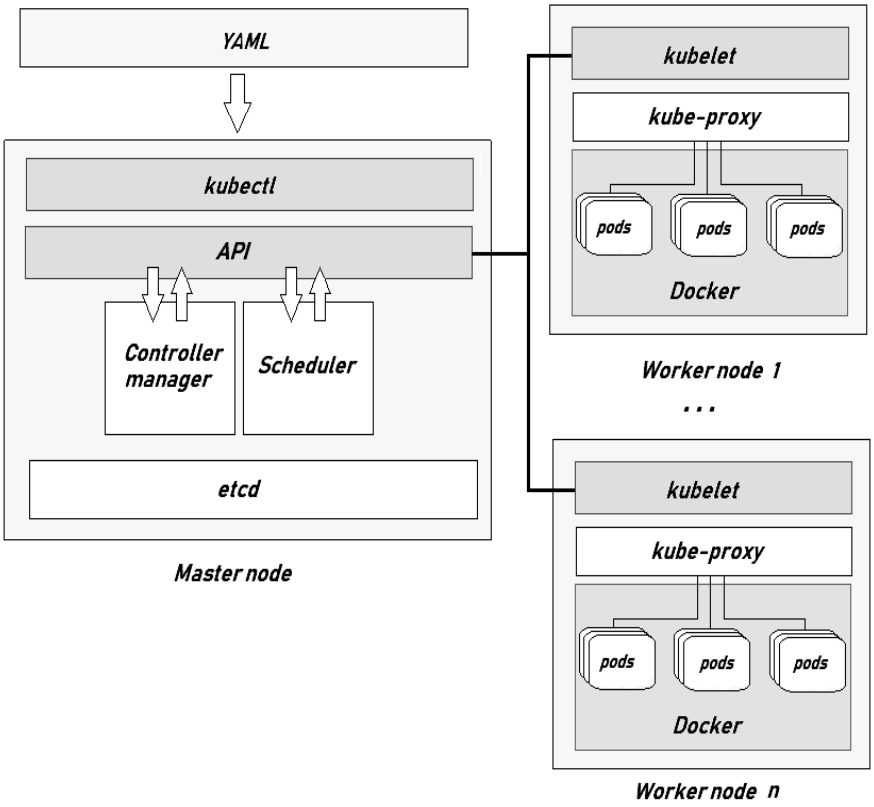


Fig. 2. Master node and Worker node architecture

The main benefits of the Kubernetes cluster. Kubernetes allows you to quickly and efficiently deploy applications, scale at runtime, integrate new features, and use only the resources you need, optimizing your resources.

Kubernetes fully supports the DevOps model and allows you to manage full application lifecycle.

The following benefits of Kubernetes can be distinguished:

– scalable. Software can be deployed via pods, deployment process can be scaled or disabled;

– monitoring. Kubernetes lets you track completed, incompleted, and unsuccessful deployments with the ability to query the status of each container;

– version control. Kubernetes allows you to update Pods which have been deployed, using newer versions of applications and revert to the previous deployment if the current version is not stable. Kubernetes provides an option to revert to an earlier version if the application deployment process fails;

– horizontal scaling. Kubernetes automatically deploys the number of application deployment replicas based on the using defined server resources (within defined limits);

– self-healing. Kubernetes has different features such as auto-reload, replication, placement, scaling, and container load balancing.

Also, Kubernetes has its own disadvantages. These include the lack of properly built documentation and the high complexity of cluster construction. Compared to other orchestration systems such as Docker Swarm, Kubernetes is more difficult to understand but more reliable orchestration systems.

Kubernetes cluster and Cloud providers

Cloud providers like Amazon (AWS), the Google Cloud Platform (GCP), and Microsoft Azure (MA) have services capable of configuring Kubernetes clusters.

Amazon Web Services. AWS is a public cloud, is the first in the public cloud market, with a huge and growing suite of available services, as well as the world's largest network of data centers [1]. AWS has its own ECS container orchestrator, but it's different from Kubernetes.

The Kubernetes Operations (kops) project has become the standard for creating, upgrading and managing Kubernetes clusters on Amazon Web Services [7].

Advantages of using Kubernetes with AWS Kubernetes is node updates, additions, and deletions.

The disadvantages of using Kubernetes with Amazon Web Services:

– lack of support for Kubernetes in the Amazon Web Services Management Console;

– no backup;

– high cost of use.

AWS uses a several payment models. You only pay for the resources and services you use. The more resources you use, the less the cost of services provided. Amazon Web Services rounding works based on hours of use [1], [7], [10].

Microsoft Azure. Microsoft Azure - Microsoft's cloud platform. Azure Kubernetes (AKS) fully managed service makes it easy to deploy and manage container applications. AKS does not include features for

updating the cluster after it is deployed. To upgrade clusters with AKS, you need to use AKS to create a new cluster, move containers to it, and remove the old cluster [10, 11].

Advantages of using Kubernetes with Azure Kubernetes:

– cluster backup is a feature of AKS;

– the Azure Management Console supports some operations with Kubernetes. You can add or delete work nodes by command in console.

The disadvantages of using Kubernetes with Microsoft Azure is the non-automated Kubernetes node upgrade.

Microsoft Azure uses a more flexible pricing system, payment for the use of cloud resources, with rounding by the minute [2].

Google Cloud Platform. Google Cloud Platform is the third in the market because it doesn't have so many different services and features as Amazon Web Services and Azure Cloud providers. However, Google offers significant scaling and load balancing. Load balancing is a major feature because Google provides a built-in general load balancer that is automatically configured when creating services.

In Amazon Web Services and Azure, the load balancer is another instance of a container that needs to be scaled.

Advantages of using Kubernetes with Azure Kubernetes:

– management of host and worker nodes is possible through the available GCP management console;

– management of master and worker nodes is possible through the available GCP management console;

– global zoom and built-in load balancer.

The disadvantages of using Kubernetes with GCP is that some features cannot be changed or configured.

The Google Cloud Platform has a similar billing system as Azure, but with a rounded usage of resources over a period of 10 minutes [2]. Google Billing also takes into account the network traffic used by the features in the aggregate.

Conclusion

The Docker Swarm and Kubernetes are two of the most popular container orchestration systems today.

Kubernetes is the most popular system that builds an efficient container system for cluster nodes, depending on current load and available services. Kubernetes is suitable for deploying infrastructure that requires a large amount of resources. It allows you to manage a large number of hosts, run multiple

Docker containers, monitor their status, monitor collaboration, perform load balancing. Kubernetes also allows you to build a big reliable system. Compared to other orchestrators, Kubernetes is the best in terms of fault tolerance.

But Kubernetes has one major disadvantage - incomplete documentation, especially for such a complex orchestration system.

Docker Swarm is the second container orchestration system.

The advantage are simplicity, the relative ease of operation and speed of development. The disadvantage is the functionality, which does not allow to build a reliable container management system.

The choice of orchestration system depends on the tasks and goals of your project. If you need a simple system, you can choose Docker Swarm. But if you want to run more container clusters with high reliability, then you need Kubernetes.

References

1. Pevnev V.Ya. “Metody obespecheniya tselostnosti informatsii v infokommunikatsionnykh sistemakh”. *Visnik Natsional'nogo tekhnichnogo universitetu KhPI*. 2015. № 51. pp. 74-77.
2. Hypervisor [online] Available at: https://en.wikipedia.org/wiki/Hypervisor.
3. Gipervizor i ego rol' v virtualizatsii [online] Available at: https://vps.ua/blog/hypervizor-and-virtualization/ .
4. Sistema upravlinnya obchislyuval'nimi resursami zastosunkiv v umovakh konteinernoї virtualizatsiї (2018). Kubernetes. [online] Available at: http://ela.kpi.ua/bitstream/123456789/25529/1/Zagorulko_magistr.pdf.
5. Docker Documentation. (2019). *Swarm mode overview*. [online] Available at: https://docs.docker.com/engine/swarm/
6. Mesosphere.github.io. (2018). *Marathon: A container orchestration platform for Mesos and DC/OS*. [online] Available at: https://mesosphere.github.io/marathon/.
7. Amazon Web Services, Inc. (2019). *Amazon Elastic Container Service – Управление контейнерами Docker – AWS*. [online] Available at: https://aws.amazon.com/ru/ecs/.
8. Docker swarm vs Kubernetes [online] Available at: https://habr.com/ru/company/d2cio/blog/349138/ .
9. A comparative analysis of two container management systems: docker swarm and kubernetes [online] Available at: http://synergy-journal.ru/archive/article2263/.
10. Sravnenie uslug oblachnykh provaiderov: Microsoft Azure, AWS ili Google Cloud. [online] Available at: http://la.by/blog/sravnenie-uslug-oblachnyh-provayderov-microsoft-azure-aws-ili-google-cloud/.
11. Publishing, Jie Xiong, (2018). *Cloud Computing for Scientific Research*. school of Electronics and Information, Yangtze University, China.

Received (Надійшла до редколегії) 14.02.2020
Accepted for publication (Схвалена до друку) 22.04.2020

Відомості про авторів / About the Authors

Пєвнєв Володимир Яковлевич – кандидат технічних наук, доцент, доцент кафедри комп’ютерних систем, мереж та кібербезпеки, Національний аерокосмічний університет імені М.Є. Жуковського «ХАІ», Харків, Україна;

Volodymyr Pevnev– Candidate of Technical Science, Associate Professor, Associate Professor of Computer Systems, Networks and Cyber security Department, National Aviation University "Kharkiv Aviation Institute", Kharkiv, Ukraine;
e-mail: v.pevnev@csn.khai.edu; ORCID ID: http://orcid.org/0000-0002-3949-3514.

Трегуб Юлія Володимирівна – магістрантка кафедри комп'ютерних систем, мереж і кібербезпеки Харківського національного аерокосмічного університету ім. М.Є. Жуковського, Харків, Україна;
Yuliia Trehub – Master's Degree in Computer Systems, Networks and Cyber Security, Kharkiv National Aerospace University. M.E. Zhukovsky, Kharkiv, Ukraine;
e-mail: j.tregub@student.csn.khai.edu; ORCID ID: http://orcid.org/0000-0003-1495-3112

Аналіз та дослідження відомих систем оркестрації для побудування мікросервісної інфраструктури

В. Я. Пєвнєв, Ю. В.Трегуб

Анотація. Предметом вивчення в статті є сучасні системи оркестрації. При розробці, для скорочення часу побудування інфраструктури та зменшення затрат на обладнання та підтримку роботи системи, існує велика потреба у застосуванні різних методів віртуалізації. Основними технологіями віртуалізації є віртуалізація на основі використання гіпервізора і контейнерна віртуалізація. Найбільш популярними системами управління контейнерами (або системами оркестрації контейнерів) є Kubernetes та Docker Swarm, обидві з яких базуються на платформі Docker. Використання однієї із них дозволяє швидше і ефективніше розроблювати додатки, стандартизує виконувані додатками операції та оптимізує використання ресурсів. Завдяки Docker користувачі отримують об'єкт, який з високою надійністю можна запускати на будь-якій платформі. Також, у статті наведено особливості спільного використання систем оркестрації разом з різними Cloud провайдерами, а також порівняння самих хмарних постачальників. **Метою** є детальний аналіз існуючих інструментів оркестраціїї контейнерів, проведення порівняльної характеристики Kubernetes і Docker Swarm, та вибір кращої із них. **Результати** порівняння Kubernetes та Docker Swarm показують, що Kubernetes – один із найкращих інструментів оркестрації. Kubernetes підходить для розгортання інфраструктури, яка потребує великої кількості ресурсів, та дозволяє обслуговувати величезну кількість хостів, запускати на них численні контейнери Docker, відстежувати їх стан, контролювати спільну роботу, проводити балансування навантаження. Kubernetes дозволяє побудувати надійну систему. У порівнянні з іншими оркестраторами, Kubernetes є найкращим з точки зору реалізації відмовостійкості. Якщо потрібне швидке налаштування і є прості вимоги до конфігурації, Docker Swarm може стати хорошим варіантом завдяки своїй простоті. **Висновки**. Сьогодні на ринку представлено дві системи оркестрації контейнерів: Docker Swarm та Kubernetes. Kubernetes – найпопулярніша система, яка вибудовує ефективну систему оркестрації контейнерів по вузлах кластеру в залежності від поточного навантаження і наявних потреб в роботі сервісів. Docker Swarm – друга, але більш проста за реалізацією система оркестрації контейнерів. Вибір системи оркестрації залежить від поставлених завдань. Якщо потрібна проста система, можна вибрати Docker Swarm. Але якщо потрібно запускати більшу кількість контейнерних кластерів, тоді потрібно використовувати Kubernetes.

Ключові слова: технології віртуалізації; системи оркестрації; Docker; DockerSwarm; Kubernetes; Ansible; Terraform; Amazon Web Services, Google Cloud Platform, Microsoft Azure.

Анализ и исследование известных систем оркестрации для построения микросервиснои инфраструктуры

В. Я. Певнев, Ю. В.Трегуб

Аннотация. Предметом изучения в статье являются современные системы оркестрации. При разработке, для сокращения времени построения инфраструктуры и уменьшения затрат на оборудование и поддержку работы системы, существует большая потребность в применении различных методов виртуализации. Основными технологиями виртуализации является виртуализация на основе использования гипервизора и контейнерная виртуализация. Наиболее популярными системами управления контейнерами (или системами оркестрации контейнеров) являются: Kubernetes и Docker Swarm, обе из которых базируются на платформе Docker. Использование одной из них позволяет быстрее и эффективнее разрабатывать приложения, стандартизирует выполняемые приложениями операции и оптимизирует использование ресурсов. Благодаря Docker пользователи получают объект, который с высокой надежностью можно запускать на любой платформе. Также, в статье приведены особенности совместного использования систем оркестрации вместе с различными Cloud провайдерами, а также сравнение самих облачных поставщиков. **Целью** является детальный анализ существующих инструментов оркестрации контейнеров, проведение сравнительной характеристики Kubernetes и Docker Swarm, и выбор лучшей из них. **Результаты** сравнения Kubernetes и Docker Swarm показывают, что Kubernetes - один из лучших инструментов оркестрации. Kubernetes подходит для развертывания инфраструктуры, которая требует большого количества ресурсов, и позволяет обслуживать огромное количество хостов, запускать на них многочисленные контейнеры Docker, отслеживать их состояние, контролировать совместную работу, проводить балансировку нагрузки. Kubernetes позволяет построить надежную систему. По сравнению с другими оркестратор Kubernetes является наилучшим с точки зрения реализации отказоустойчивости. Если требуются быстрые настройки и требования к конфигурации просты, то Docker Swarm может стать хорошим вариантом благодаря своей простоте. **Выводы**. Сегодня на рынке представлены две системы оркестрации контейнеров: Docker Swarm и Kubernetes. Kubernetes - самая популярная система, которая выстраивает эффективную систему оркестрации контейнеров по узлам кластера в зависимости от текущей нагрузки и имеющихся потребностей в работе сервисов. Docker Swarm - вторая, но более простая по реализации система оркестрации контейнеров. Выбор системы оркестрации зависит от поставленных задач. Если нужна простая система, можно выбрать Docker Swarm. Но если нужно запускать большее количество контейнерных кластеров, тогда нужно выбирать Kubernetes.

Ключевые слова: технологии виртуализации; системы оркестрации; Docker; DockerSwarm; Kubernetes; Ansible; Terraform; Amazon Web Services, Google Cloud Platform, Microsoft Azure.