Рисунок 7 – Деление картины «Мадонна с младенцем, шестью ангелами и Иоанном Крестителем» золотым пятиугольником в приложении GeoGebra

Разработанные задания позволяют учащимся приобрести навык работы в приложении GeoGebra, освоить математические расчеты по правилу Золотого сечения, повысить уровень знаний о сферах применения правила Золотого сечения.

Список использованных источников

1. Сопроненко Л. П., Григорьева Я. М. Анализ золотого сечения с помощью средств компьютерной графики. Университет ИТМО, 2015. – 50 с.

2. Трушникова, М. Правило золотого сечения для создания гармоничной картины [Электронный ресурс] – Режим доступа: https://izo-life.ru/pravilo-zolotogo-secheniya.html – Дата доступа: 25.10.2022.

3. Шевелев, И. Ш., Марутаев М.А., Шмелев И.П. Золотое сечение : Три взгляда на природу. Москва : Стройиздат, 1990. – 339 с.

4. Щукина, Г.И. Активизация познавательной деятельности учащихся в учебном процессе. М.: Просвещение, 1979. – 160 с.

UDC 004.94

TEACHING CONTAINER VIRTUALIZATION AND THE BASICS OF THE KUBERNETES CONTAINER ORCHESTRATOR

D.A. Kostiuk, P.N. Lutsiuk
Brest State Technical University, Brest, Belarus, d.k@list.ru

The experience of developing a course on the basics of using the Kubernetes container orchestrator is presented. The course includes the principles of container virtualization, architectural features and key components of the Kubernetes platform, deploying and scaling applications based on it. The ways to overcome the high input threshold of Kubernetes are discussed. The structure of the theoretical and practical parts of the course, measures to ensure the operability of the studied software in an isolated network segment are given.

Containerization of applications is an essential part of the current approach to systems engineering, which involves the integration of the software systems development and operation when building complex applications. It is based on the use of a complex of such modern software technologies as virtualization, automatic audit and performance control, solving problems using heterogeneous systems that include several different platforms (for example, GNU/Linux and Microsoft Windows).

One of the increasingly popular platforms to implement this approach is Kubernetes – an open-source, complex container orchestration system developed by Google, that ensures the performance of containerized applications.

Kubernetes objects deploy and scale applications based on their memory, CPU, and other requirements. Any container virtualization system that complies with the Open Container Initiative (OCI) is supported. Kubernetes provides scaling and load balancing, automatic service discovery, and secrets management. Containers of one or more applications are isolated from each other until the developer/devops engineer decides to connect them, allowing more important applications to be used without the risk of interference. The platform takes care of the technical complexity of transparent container interaction and service auto-discovery by providing the appropriate capabilities through the API.

The scope of the Kubernetes usage is determined by the following reasons:

- the platform is very well suited for scale-in/scale-out systems and stateless systems;
- it is popular as a foundation for microservices;
- it proposes a number of benefits being used as a basis for various kinds of clusters that need reachability, auto-discovery and/or self-repair.

A problem that is regularly encountered when learning Kubernetes is a high barrier to entry. On the one hand, the reason is a fairly large set of entities of this platform, thanks to which the Kubernetes-based cluster acts as a single unit; on the other hand, a significant number of software components which the platform includes (and some of which are completely interchangeable) plays the role. In fact, a cluster is functioning as a single organism (automatically performing updates, scaling and self-healing) and therefore has rather high internal complexity.

For this reason, despite the high demand, studying Kubernetes by students often turns out to be more of an experiment.

It is also necessary to mention the difficulties of presenting its theory in languages other than English. The platform's documentation uses a large number of new concepts referenced with capitalized words to emphasis their specific meaning different from those outside of Kubernetes. The problem is well illustrated by the poor adaptability to the Russian language: it is obvious that there are currently no well-established Russian terms for the platform, and in some cases, a literal translation turns out to be the worst option. A typical example is the term "Deployment", which in Kubernetes denotes not only the process of deploying software, but also a cluster object responsible for deploying a component. The literal translation is unsuccessful in the case of a group of objects, at least because in Russian this word does not occur in the plural. The solution used in our course presented here was to use non-translated terms capitalized as in original documentation.

In our case, a course was developed to study Kubernetes by senior students who are already familiar with a number of necessary technologies, including computer networks, elements of GNU/Linux system administration, and development of client-server applications [2].

The developed course is designed for the first stage of higher education, and involves further continuation within the second stage.

The theoretical part of the first-stage course covers the concepts of Node (a separate computer or virtual machine on which containers are running), Pod (a separate unit containing at least 1 container, that the Kubernetes Scheduler operates to start any work), Replication Controller (whose task is to maintain a given number of copies of the Pod according to the Label Selector), as well as Service (an entry point that gives the client application access to the Pods it needs). Less detailed coverage was chosen for Load balancer, Secrets (objects to securely store passwords, encryption keys and other similar sensitive data), Probes (which are the way for Kubernetes to detect malfunction of the cluster components), isolation with Namespace, influencing Pods to be created on specific nodes with taints/tolerations and affinity/antiaffinity, and using Volumes to create stateful applications.

The practical part of the course was built on the basis of Minikube, a specialized Kubernetes distribution designed to be deployed on a local machine with a combination of hardware and container virtualization. Minikube has a number of limitations related primarily to its local nature. However, the practical part of the course, due to time constraints, does not affect a number of aspects of the platform that are incompatible with it; topics such as, for example, the implementation of affinity and anti-affinity are covered only by the theoretical part.

One of the tasks needed to be solved for the laboratory workshop included in the course was the functioning of Minikube in an isolated segment of the local network, with limited access to external Internet resources (the reason for the restriction was both saving external traffic and considerations of the internal security policy of the local network with personalized Internet access via VPN).

The need for good Internet access is caused by two reasons:

- Minikube downloads a ready-made installation image with Docker and Kubernetes components, as well as the latest versions of the kubelet and kubeadm tools;
- When deploying applications to a Kubernetes cluster, application containers are downloaded by the Docker virtualization system.

Two options for using Kuberbnetes with limited external access were considered and tested:

- creating a private Docker registry and reconfiguring the system to use it exclusively, as well as deploying a private image repository for Minikube;
- running Minikube and deploying educational applications with Internet access enabled, followed by cloning the resulting profile to classroom workstations: this option solves both deploying images for Minikube and importing Docker images through the file system instead of getting them over the network.

In practice, after experimenting with both options, we settled on the second one, as the least time-consuming (taking into account the existing image replication system for workstations) [3, 4].

The structure of the developed practical workshop for students of the first stage of higher education includes 4 laboratory works [1]. The first work is exploratory in nature: it talks about Minikube and virtualization systems that it can use on various hardware platforms, as well as installation features if used on personal devices. The second one discusses the features of accessing a Kubernetes cluster using the command line (kubectl), including minimal access configuration (kubectl proxy), and a web interface (Kubernetes Dashboard). The third one deploys a minimal web application to the cluster, including checking the created Deployment, ReplicaSet and Pods, as well as the Service accessed via NodePort. The fourth work allows students to study the deployment of a multi-component application with a prepared client part based on web technologies and a server subsystem based on the MongoDB database management system.

References

1. Касцюк Д.А., Луцюк П.М. Практычнае вывучэнне сродкаў кантэйнернай віртуалізацыі і платформы Kubernetes // Десята науково-практична конференція FOSS Lviv 2021: Збірник наукових праць. – Львів, 17-19 червня 2021 р. – С. 19–21.

2. Костюк Д.А., Ильяшевич Д.А. Опыт внедрения свободного ПО в учебный процесс для специальностей информатики и радиоэлектроники // Свободное программное обеспечение в высшей школе: тезисы докладов III Конференции., 2–4 февраля 2008 г. – Переславль-Залесский, 2008. – С. 48–51.

3. Коваленко В.Ю., Костюк Д.А. Виртуализованная ферма для тестирования и демонстрации приложений платформы Android с веб-доступом // Вестник Брестского государственного технического университета. 2015. №5 Физика, математика, информатика - С. 45-48.

4. Пойта П. С., Костюк Д. А., Дереченник С. С., Луцюк П. Н. Повышение сетевой безопасности в компьютерном парке вуза за счет буферизации и изоляции ресурсов // Электроника инфо. – 2013. – No.6 (96). – С. 111–113.

УДК 004.9

ТЕХНИКА ДИНАМИЧЕСКОЙ ВИЗУАЛИЗАЦИИ В ЗАДАЧЕ ОРГАНИЗАЦИИ ОБУЧЕНИЯ В ОБЛАСТИ КОМПЬЮТЕРНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

А.Ю. Савицкая
Брестский государственный технический университет, Брест, Беларусь,
stendur6@gmail.com

The features of the application of dynamic visualization techniques in the construction of training systems in the field of computer information technologies are considered. The key stages of a systematic approach to the design and development of such systems are given.

Одним из приоритетных направлений в области повышения качества обучения техническим дисциплинам является разработка и внедрение инновационных образовательных технологий, основанных на применении современных аппаратно-программных средств компьютерной техники. Практика применения компьютерных обучающих систем совместно с традиционными средствами