Theoretical and Applied in Automating Kubernetes Resources

Pavel-Cristian CRACIUN[1], Cristian Robert NECULA[2]
[1]Bucharest University of Economic Studies, [2]Oracle Romania
craciunpavel18@stud.ase.ro, cristian.necula@oracle.com

This paper represents both a theoretical and practical approach to the possibility of managing and automating Kubernetes resources through the lens of a web solution. The solution proposed, KubeGen, represents a graphical user interface, template-based resource generation and built-in validation. We demonstrate how KubeGen can streamline Kubernetes resource creation by enhancing compliance with best practices. By highlighting critical pain point in Kubernetes resource management, the solution offers a more efficient and user-centric method for deploying and maintaining modern applications.
Keywords: *Kubernetes, cloud infrastructure management, DevOPS*
DOI: 10.24818/issn14531305/27.2.2023.04

1 Introduction

The approach in which software is written, distributed, and maintained has undergone a transformation thanks to the widespread adoption of Kubernetes as the standard orchestration platform for containerized applications. Scaling, deploying, and supporting applications have been made easier and more automated with Kubernetes' powerful resource management and allocation abstractions. [1] However, creating Kubernetes resource configuration files can be challenging for developers and system administrators, especially for complicated applications. It would also be time-consuming for the inevitably error-prone user to keep up with the latest best practices, discover deprecated resources, and integrate observability tools. For this fact, the presented solution "KubeGen", a complete web-based tool that simplifies the production and management of Kubernetes resources, is highlighted to overcome these difficulties.

Using an easy-to-use Graphical User Interface, KubeGen allows users to quickly generate, modify and manage Kubernetes resources, including Deployment, Pods, Ingress, Service, StatefulSets, etc. KubeGen reduces the risk of misconfiguration and ensures adherence to accepted best practices by providing a guided approach that leverages the power of predefined templates and automatic validation. One of the distinguishing features of the solution is the use of Pluto, an open-source tool for identifying deprecated Kubernetes API versions. This solution integration allows users to automatically verify that their resource configurations comply with the latest Kubernetes API standards, preventing potential deployment issues. Furthermore, KubeGen offers seamless integration with well-known monitoring and observation tools such as Prometheus and Grafana. It also enables users to quickly include monitoring and alerting capabilities in their Kubernetes resource configurations, providing vital information about the functionality and health of their applications.

The concept and implementation of KubeGen is outlined in this paper, highlighting its various features and the advantages it offers to users. Also included are several real-world usage examples that show how KubeGen can be used to accelerate the creation and management of Kubernetes resources for a variety of applications and markets.

2 Challenges in management's infrastructure creation

Throughout the years, more and more organizations have chosen to apply everything through a DevOps and Agile perspective in order to deliver a product with an efficient velocity, in order to satisfy a client demands. Agile concepts place a strong focus on interpersonal communication, making businesses more adaptable to change, and strongly encouraging consumer engagement. [2]

DevOps, a bridge between two teams (Development and Operations), appears in the landscape with the goal of accelerating the software delivery speed, even if embracing an iterative and incremental development approach, as in agile techniques, helps firms fulfill constantly evolving client expectations. [3]

These two methodologies with their own set of practices assures any organization that software delivery will bloom alongside with people management. In simple words, both mentioned key factors highlight common pillars, for instance Continuous Integration and Collaboration, Continuous Delivery. [4]

With the constant growth of this production sphere, power comes with great responsibility. Multiple strategic concerns have appeared in most businesses in which cost optimization represents the main goal in order to not sacrifice product quality or usefulness. In hopes of sustaining the appropriate levels of profitability, businesses must have effective cost management systems and be ready to adopt cost reduction programs. Many businesses carry out cost-cutting measures as projects that enlist the required departments and staff to work toward a predetermined target cost or cost-cutting objective. [5]

As presented in Introduction, the infrastructure of an application plays a crucial part in its overall success and performance. Applications that run smoothly and efficiently which grants users the best possible experience describes a robust infrastructure whereas on the other side, user demands, traffic or data processing needs, require an infrastructure to be scalable as the product expands.

Therefore, in this new challenging space, many organizations start thinking to reshape their IT strategy in order to increase business value and agility, accelerating business transformation and innovations.

Kubernetes, a tool based on containerization, maintaining and deploying applications, represents on being one of the most important aspects in the creation of the infrastructure foundation. The only flaw which is encountered in this sphere is the manual and traditional approach that highlights the problems for which the future solution is based on:

- Time-consuming manual creation of YAML (YAML Ain't Markup Language) files, in which the individual has to create them from scratch using different code editors.
- Increased time spent on troubleshooting and resolving deployment issues.
- Less efficient collaboration among team members, which leads to potential miscommunication.
- Lack of capability to reuse assets for new deployment in a fashion way.
- High level of effort to maintain and administrate overall infrastructure.

This is where the proposed solution will take place. An approach for companies that focuses on developing software products. Time, cost optimization and team collaboration might be improved through "KubeGen" as shown:

- Streamlined, automated generation of YAML files through an intuitive UI.
- Faster resolution of deployment issues due to application's validation and error-checking capabilities.
- Enhanced collaboration, enabling team members to share, plan and review configurations.

2 Design and Architecture

The construction of a Kubernetes resource automation tool is examined in this session, with a particular emphasis on resource production utilizing customizable templates, built-in validation methods, and impairment detection. It also emphasizes how modern monitoring and observability tools may be seamlessly integrated with container orchestration systems to improve overall resource management.

2.1 Generation of resources based on template

The project is made in Spring Boot with Java 17 (Backend) and ReactJs (Frontend). A schematic representation of the architecture can be seen in Fig. 1. The backend is structured according to the Domain Driven Design Paradigm. [6] Thus, the functionality and structure

of the packages revolves around the Domain, in our case template.

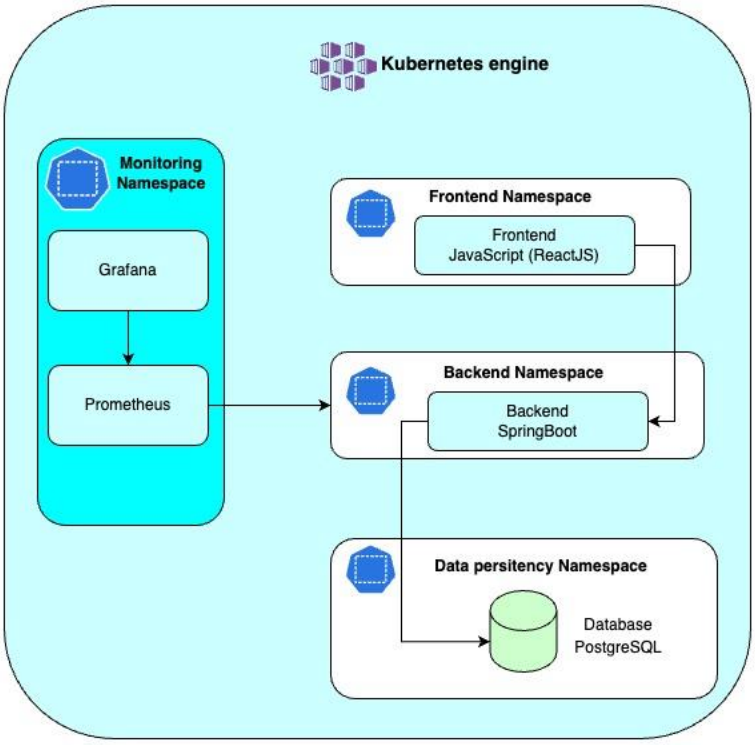


Fig. 1. KubeGen architecture

Thus, the package structure is as follows, Figure 2:

1. Data where we will keep the Repository type objects:
   - InfrastructureTemplateRepository
2. Domain, which will host Domain and Enum type objects:
   - ComponentKind
   - InfrastructureTemplate
   - ProvisionedPath
3. Resource, where the Rest type resources will be kept:
   - ArchiveResource
4. Service, which will contain the business logic of the application, through Service objects.

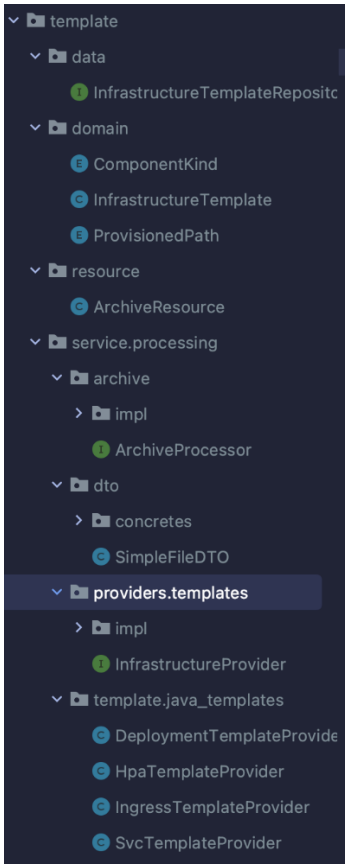


Fig. 2. Package Structure

In the frontend part, we use React during implementation which will give us flexibility in the project's structure. Project structure is as follows. The components package includes all reusable application components, such as *CustomButton*, *CustomHeader*, and *Modal*. Modules for applications are included in the package of modules:

- **InfrastructureStepper**: This stepper component, which specifies the steps that a user must complete within an application, refers to each component that will serve as a step, collect completed data, and send it to the backend.
- **DeploymentStep**: This component serves as the deployment entry point; it is a form that requires user completion of all fields in order to define a deployment in Kubernetes.
- **ServiceStep**: This component acts as the service's entry point; it is a form that the user must fill out in order to define a service in Kubernetes.
- **IngressStep**: This component serves as the Ingress entry point; it is a form that requires user completion of all fields in order to define an Ingress in Kubernetes. Dates are stored in the React state and implemented with Redux (helps to write application that have a consistent behavior, which runs in environments such as client and server) [7]. In order to decrease costs, data will be sent to the backend through HTTP requests.

The application logic works as follows, a user calls the /archive/generate API, along with which it will send an InfrastructureDTO object containing the fields filled by the user in the GUI. This is forwarded to a service called *ArchiveProcessor*, responsible for writing the archive and files needed by the infrastructure, via the *byte[] createArchiveFromFiles* method. This method will in turn call another component, a simple bean annotated with @Component that will handle the creation of *SimpleFileDTO* objects.

These objects represent the very files that must be written to the archive, and which will

be created one by one in the following way. An *InfrastructureTemplate* domain will be fetched from the database depending on the type of resource generated (e.g., Deployment) and will contain the template of each file type. Thus, once brought from the database, the template is processed, and the fields specific to each Kubernetes component will be replaced with the values sent by the user from the graphical interface. Further, these templates are transformed into *SimpleFileDTO* files and parsed, one at a time, by the *byte [] createArchiveFromFiles* method. At the end, a stream of bytes is returned to the user, which will be passed by the frontend and which, in the end, will reach the user in the form of a zip archive.

2.2 Validation and built-in impairment detection

Pluto, a particular validation tool, is used to verify Kubernetes component version updates. It is set up on the Kubernetes cluster for the application and operates in the background. The API version is passed to the Java service, which makes a call to Pluto, validates the version, and then returns the result to the frontend, which displays it appropriately to the user.

2.3 Integration with monitoring and observability tools

The connection with Grafana and Prometheus is accomplished by deploying a sidecar container along with the application image that has a Prometheus operator that will scrape the metrics and then transmit them to the main Prometheus instance. The application will also provide a YAML file with the configurations for the primary instances of Prometheus and Grafana, as well as a script to install the instances on the Kubernetes cluster.

3 Kubernetes Resource Automation Solution

The solution proposed by this work will take the form of a web application that will allow in an optimal and efficient way the generation, monitoring and management of Kubernetes resources, the aim is to offer the user an easier alternative regarding the creation of the desired resource avoiding one of the most common "*CrashLoopBackOff error*" errors. This is highlighted when a pod is active but one of its containers keeps resuming due to termination. As a result, the container keeps entering the *Start-crash-start-crash* loop. This error can have several causes but the most common is a simple configuration file write error. [8]
The application is built around 2 types of users:

- The web platform visitor is not authenticated in the application and only has access to general information. It cannot create Kubernetes resources.
- Registered user: can create, manage and monitor Kubernetes resources to help them build an infrastructure for their application.

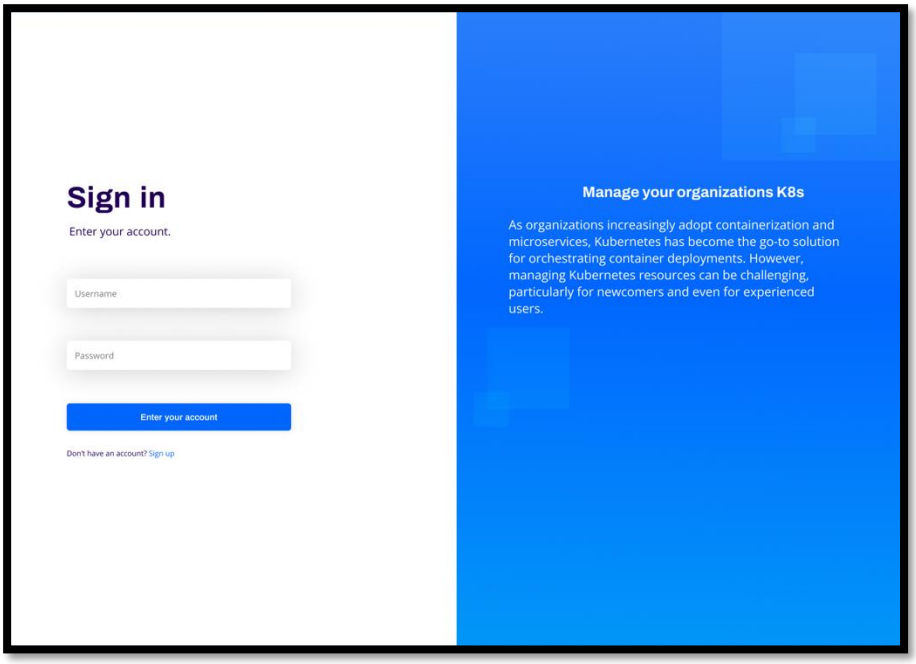


Fig. 3. Login page

In the Sign in part (Fig. 3), the user can log in using the Username and Password of the previously created account. If he has not created the account, he can create it by pressing the Sign-up link, where he will be redirected to a form.

Fig. 4 shows the page that provides the user with a list of Kubernetes resources previously created by him. On this list, the user can modify resources, view them, check them with the Pluto tool, or apply notes for future decisions.

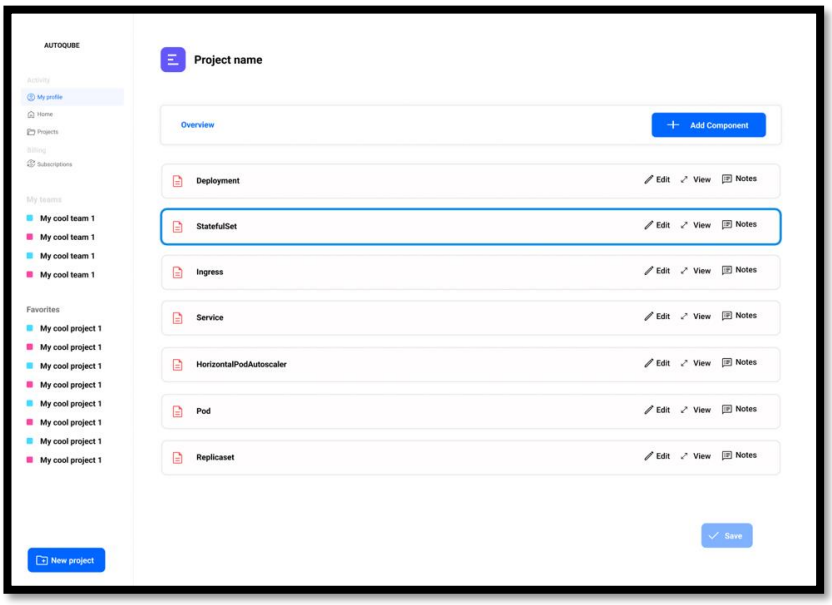


Fig. 4. List of Resources

The resource creation procedure is started when the user navigates to the resource list UI page and presses the "+" button. The user is prompted to select the required Kubernetes resource type (e.g., deployment) from a dropdown menu in a pop-up window (Fig. 5).

The program displays a form with the necessary information that the user must fill in depending on the type of resource he has chosen. These attributes align with the specific configuration options for the selected resource.

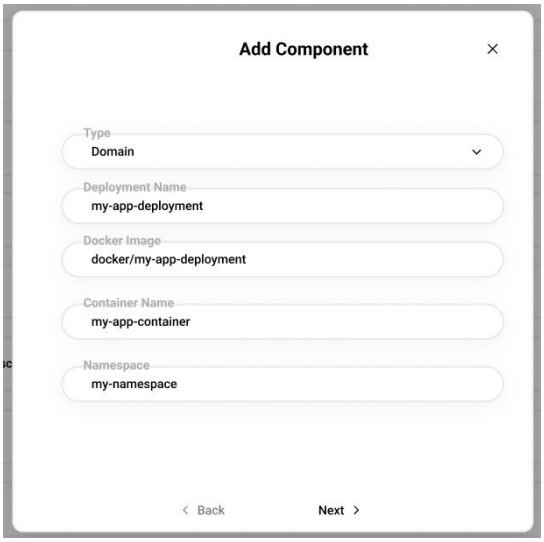


Fig. 5. Create Resource

The user can review the configuration produced for deprecating the API version using the built-in Pluto tool. The user can make the necessary changes to ensure compliance with the latest Kubernetes API standards if issues are found. The resulting YAML file, which is now ready to deploy to the user's Kubernetes cluster, can be downloaded if the user is satisfied with the resource configuration.

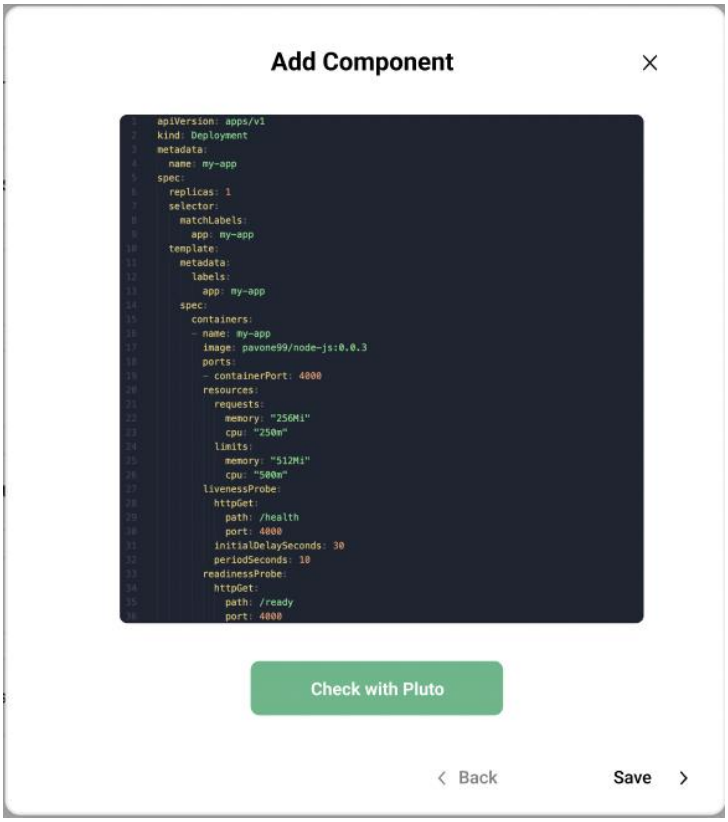


Fig. 6. Pluto checker

At the end of these steps the created resource will be highlighted (Fig. 7)) and the user will be able to run the "*kubectl apply -f resource.yaml*" command to pick up the desired infrastructure for the related application (Fig. 7).

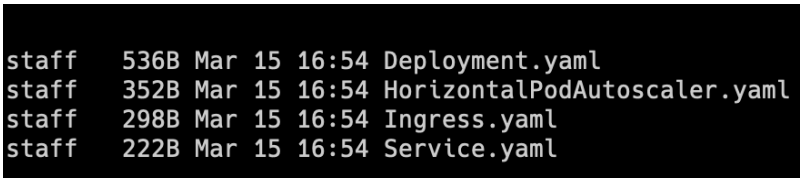


Fig. 7. List of resources created

Fig. 8. Applying resource

4 Effort, cost and future work

As we discussed in "Challenges in management's infrastructure creation" chapter, for this sphere it is essential to analyze the costs and efforts that implies the need for using such tool.

From the cost perspective, as Kubernetes represents an open-source container orchestration engine it will be natural for this solution to be the same as it takes its roots from the official Kubernetes documentation [9]. The training effort for the user is to focus on getting familiar with the web interface for being able to use it while having less experience in orchestrating containers using Kubernetes.

KubeGen from a personal point of view would provide to be useful in streamlining Kubernetes resource management.

There are still several opportunities for future enhancements and additions that could improve functionality and user experience further:

- Support for more Kubernetes resources will allow users to handle more complicated situations and applications, extending KubeGen's capabilities to encompass a wider variety of Kubernetes resources and settings.
- System integration with version control: users could easily manage and monitor changes in their resource settings over time if there was a seamless connection with well-known version control systems such as Git.
- Automated resource optimization: Kubernetes installations could become more efficient by using machine learning techniques or heuristics to monitor resource consumption trends and provide recommendations for improving resource allocations [10].
- Improved Teamwork Features: Users can work more efficiently in a team environment by introducing tools that promote teamwork, such as configuration sharing, comments and change tracking.
- Integrating in a homogenously way monitoring and observability to benefit from Prometheus and Grafana, so that every infrastructure to be fully monitored and managed.
- Support for multi-cluster management would help enterprises with more complicated infrastructure requirements by extending KubeGen to handle resource management across multiple Kubernetes clusters. KubeGen could further establish itself as a comprehensive and easy-to-use solution for creating and managing Kubernetes resources by investigating and implementing these future updates [11].

5 Conclusion

KubeGen represents an entirely web-based tool that would make it simple to create and manage Kubernetes resources. By offering a simple GUI, KubeGen will make it easy for both new and expert users to generate and

configure Kubernetes resources with confidence. Because to its ease of use, Kubernetes has a shorter learning curve, and the deployment process moves faster. KubeGen's use of template-based resource creation guarantees that users adhere to generally acknowledged best practices, and the built-in validation and interaction with Pluto lessens the likelihood of using obsolete APIs and wrong settings.

Collectively, these enhancements increase the stability and dependability of Kubernetes resources. KubeGen's commitment to enhancing the user experience is further evidenced by its simple interface with popular observability and monitoring tools like Prometheus and Grafana, which offers features that allow user to focus on the optimization of Kubernetes infrastructure components. By making the integration of monitoring and alerting capabilities simpler, KubeGen will drive users to maintain a clear view of their application's performance and health.

Overall, KubeGen is successful in addressing a number of critical issues with the Kubernetes resource generation and management process, giving users a more streamlined and effective way to deploy and maintain containerized applications.

As the usage of Kubernetes grows, tools like KubeGen will become more and more important for enhancing the deployment and administration of contemporary applications.

References

[1] J. Beda,K Hightower and B.Burns, *Kubernetes: Up and Running*, O'Reilly Media, September 2017, ISBN: 9781491935675;

[2] M. Younas, D. N.Jawawi, I. Ghani, T. Fries, and R. Kazmi, "Agile development in the cloud computing environment: A systematic review", *Information and Software Technology*, no. 103, pp. 142 –158, 2018), ISSN: 0950-5849. DOI: https://doi.org/ 10.1016/j.infsof.2018.06.014 Available: http://www.sciencedirect.com/science/article/pii/S0950584918301319.

[3] A. Krishna Kaiser, "Introduction to DevOps", *Reinventing IT for the Age of DevOps: Innovative Techniques to Make Processes Agile and Relevant*, Berkeley, CA: Apress, pp. 1–35, 2018, ISBN: 978-1-4842-3976-6. DOI: 10.1007/978-1-4842-3976-6_1, https://doi.org/10.1007/978-1-4842-3976-6_1.

[4] G. Benguria, J. Alonso, I., Etxaniz, L. Orue-Echevarria and M. Escalante, "*Agile Development and Operation of Complex Systems in Multi-technology and Multi-company Environments: Following a DevOps Approach*". In: European Conference on Software Process Improvement. Springer, pp. 15–27, 2018.

[5] D. Machado, A. Tereso, and P. Afonso, "Ratio Project Planning: cost optimization projects in the production phase". *Procedia Computer Science*, 219, 2023, https://doi.org/10.1016/j.procs.2023.01.501.

[6] E. Evans, "Domain-Driven Design: Tackling Complexity in the Heart of Software", Addison-Wesley Professional, 2004.

[7]. Redux.js, "Getting Started with Redux", Available: https://redux.js.org/introduction/getting-started

[8] M. Perry, "*7 Common Kubernetes Pitfalls*", September 3, 2022. Available: https://www.qovery.com/blog/7-common-kubernetes-pitfalls

[9] Kubernetes, https://kubernetes.io/docs/home/.

[10] B. Likosar, (2022), "*Using Machine Learning to Automate Kubernetes Optimization*". Available: https://thenewstack.io/using-machine-learning-to-automate-kubernetes-optimization/.

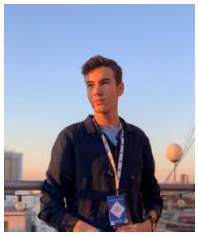
[11] S. Burns, "*Strategies for Kubernetes multi-cluster management*", 2023. Available: https://www.techtarget.com/searchitoperations/tip/Strategies-for-Kubernetes-multi-cluster-management.

[12] K. Chasioti, "*BizDevOps: A process model for the Alignment of Devops with BusinessGoals*", Utrecht University, 2019. Available: https://studenttheses.uu.nl/handle/20.500.12932/33534.

[13] J. Cleland-Huang, "Traceability in Agile Projects", *Software and Systems Traceability*. Ed. by J. Cleland-Huang, O. Gotel,

and A. Zisman, London: Springer London, 2011, 265-275. ISBN: 978-1-4471-2239-5. DOI: 10.1007/978-1-4471-2239- 5_12. Available: https://doi.org/10.1007/978-1-4471-2239-5_12.

[14] Razorops, "*Kubernetes-The de facto standard to deploy and operate containerized applications*", June 3, 2022. Available: https://www.linkedin.com/pulse/kubernetes-the-de-facto-standard-deploy-operate-containerized-?trk=organization-update-content_share-article.



Pavel Cristian CRĂCIUN has graduated the Faculty of Economic Cybernetics, Statistics and Informatics in 2021. Currently he is enrolled in a Master programme on Informatics Systems for the Management of Economic Resources from the Academy of Economic Studies. He is a DevOps in the field of engineering. His work focuses on the developing, maintaining and observing of software applications.

Cristian Robert NECULA has graduated the Faculty of Economic Cybernetics, Statistics and Informatics in 1999. Experienced Technology Architect with a demonstrated history of working in the information technology and services industry. Strong information technology professional skilled in Oracle Database, Databases, Middleware, DevOps, Datawarehouse, Blockchain, Solution Architecture, and Software Project Management.