

# Automatic Deployment to Kubernetes Cluster by Applying a New Learning Tool and Learning Processes

Tomáš Golis<sup>1,\*†</sup>, Pavle Dakić<sup>1,2,\*†</sup> and Valentino Vranić<sup>1</sup>

<sup>1</sup>*Institute of Informatics, Information Systems and Software Engineering, Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Ilkovičova 2, 842 16 Bratislava 4, Slovakia*

<sup>2</sup>*Faculty of Informatics and Computing, Singidunum University, Danijelova 32, Belgrade, Serbia*

## Abstract

The rapid growth and widespread adoption of containerization technologies, such as Docker, and the increasing popularity of Kubernetes as a container orchestration platform have significantly shaped the landscape of modern software development and deployment. Containers have created a paradigm shift in application packaging, offering a consistent and portable environment across diverse infrastructure setups. Meanwhile, Kubernetes has emerged as the dominant choice for managing containerized applications at scale, providing features such as automated scaling, load balancing, and fault tolerance. However, despite the numerous advantages containers and Kubernetes offer, deploying applications to a Kubernetes cluster often poses challenges due to the steep learning curve and complex configuration requirements. To address these challenges, this paper aims to introduce a comprehensive learning tool that automates and simplifies the deployment process on Kubernetes clusters.

## Keywords

Kubernetes, Container orchestration, Docker image, CI/CD

## 1. Introduction

The primary objective of this research is to develop an intelligent system that simplifies and streamlines the deployment workflow, allowing users to provision their applications on Kubernetes effortlessly. Within this work, theoretical and practical aspects have been considered with the possibilities of effective application within the academic and professional environment. The proposed learning tool harnesses machine learning techniques, including natural language processing and automated reasoning, to provide a user-friendly interface and automate various deployment tasks. It encompasses a range of advanced features, such as intelligent application packaging, automatic resource allocation, and configuration optimization, all aimed at optimizing the deployment process which can be applied in many different industries [1, 2, 3, 4].

The evaluation results demonstrate that the learning tool effectively minimizes the time and effort needed to deploy applications to Kubernetes clusters. By automating repetitive and

---


*SQAMIA 2023: Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, September 10–13, 2023, Bratislava, Slovakia*


\*Corresponding author.

†These authors contributed equally.

✉ [golistomas51@gmail.com](mailto:golistomas51@gmail.com) (T. Golis); [pavle.dacic@stuba.sk](mailto:pavle.dacic@stuba.sk) (P. Dakić); [vranic@stuba.sk](mailto:vranic@stuba.sk) (V. Vranić)

ORCID [0000-0002-4258-6383](https://orcid.org/0000-0002-4258-6383) (T. Golis); [0000-0003-3538-6284](https://orcid.org/0000-0003-3538-6284) (P. Dakić); [0000-0001-9044-4593](https://orcid.org/0000-0001-9044-4593) (V. Vranić)

 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

error-prone tasks, developers and system administrators [5, 6] can dedicate more attention to application development and innovative endeavors. By bridging the divide between developers and Kubernetes, the learning tool plays a crucial role in promoting the broader adoption of containerization technologies, ultimately expediting the pace of software delivery.

The learning tool developed in this research aims to provide a seamless and efficient deployment experience for applications on Kubernetes clusters. One of its key objectives is to automate the deployment process for any application that includes a Dockerfile in its Git repository. By leveraging the Dockerfile, the tool can extract the necessary information about the application's dependencies and build an appropriate container image. This automation eliminates manual containerization intervention, saving developers and system administrators valuable time and effort.

Furthermore, the learning tool generates Helm charts that can be used by the package manager for Kubernetes, enabling the easy installation, upgrading, and deletion of applications on a Kubernetes cluster. The Helm charts generated by the tool encapsulate the application, its configuration, and any required dependencies, providing a reproducible and standardized deployment process [7]. This ensures consistency and simplifies the management of applications across different environments and Kubernetes clusters.

In addition to creating Helm charts, the learning tool integrates with a Continuous Integration/Continuous Deployment (CI/CD) pipeline [8]. This integration automates applications' building, testing, and deployment, providing a streamlined workflow from development to production. The tool sets up the necessary pipeline configurations, incorporating steps for building the container image, running tests, and deploying the application to the Kubernetes cluster using a popular CI / CD system, while theoretical analysis was discussed and defended in our previous publications [9, 10].

Applying an operation automation approach can be very useful for a pragmatic engineer who wants to learn as soon as possible and be effective in his work without the limitations that arise during learning and finding the appropriate directly applicable instructions. The application of this tool helps to solve errors during cluster setup and provides a process that can be repeated very little and monitored through different control panels and dashboards [11].

The work is organized in the following way using sections: Materials and Methods, Results and Conclusions.

## **2. Materials and Methods**

Within this research work, we try to apply practical methods with partial processing of numerical data collected inside the cluster itself. The material was created on the basis of the research carried out together with the cloud cluster and the connection of obtained information within the laboratory environment. the cluster itself. The material was created on the basis of the research carried out together with the cloud cluster and the connection of obtained information within the laboratory environment. The main contribution refers to the presentation of the obtained results and tests carried out within the specified environment. Basic statistical methods were used to display the results in the form of relevant graphs and data presentation through a tabular display.

## 2.1. Motivation

The underlying motivation driving this paper is to empower students and developers by providing them with a platform to enhance their understanding of the deployment process. By engaging with this platform, they can actively participate in the deployment of their applications while also gaining valuable experience in customizing the deployment process to meet the unique requirements of their applications. This hands-on approach encourages learning through practical implementation and fosters a deeper understanding of the intricacies involved in deploying applications effectively. Ultimately, our goal is to equip students and developers with the knowledge and skills to navigate the deployment landscape confidently and proficiently.

## 2.2. Selection Criteria, Keywords, and Databases

During our exploration of container orchestration technology and the examination of different principles for automatic deployment, we identified specific keywords to guide our literature search:

- Kubernetes in the cloud
- Learning CI/CD with Kubernetes
- Using Docker image for cloud computing in clusters
- Container Orchestration in Public and Hybrid Cloud

The resources used in this research paper were sourced from public scientific research databases, as well as technology-related websites, focusing on the specified keywords. These materials were collected within the time frame of 2018 to 2023 for the purpose of writing this article.

## 2.3. Literature Review

To better realize the process of creating tools for learning the DevOps field and other related technologies, related works that deal with the problem of learning and transferring knowledge to users were reviewed. By incorporating these authoritative sources, this research builds on the existing body of knowledge in the field, ensuring a solid foundation and contributing to the academic discourse surrounding the topic. The research was carried out based on the findings and insights of two notable publications [9, 10]. These findings have been referenced in the Introduction, highlighting their relevance to the research topic. In paper [10], the authors dive into specific aspects of the subject matter, providing valuable analysis and observations. Similarly, the authors [9] offer a comprehensive exploration of related concepts, shedding light on key considerations and potential solutions [12]. The authors of the [13] study employ a methodology that helps developers tackle the Fog Service Placement Problem. This problem involves finding the best possible mapping between IoT applications and computational resources. On the other hand, the authors [14] propose a deterministic variant of Local Interpretable Model-Agnostic Explanations (LIME), a popular technique used to enhance the interpretability and explainability of black-box Machine Learning (ML) algorithms. Naturally, this requires

a robust algorithm capable of accurately reconstructing road networks from satellite images, which can then be utilized as training samples.

Understanding the complexity of resource distribution and hardware load also means looking at systems from the field of Wireless Sensor Networks (WSNs) for later direct physical use. In this connection, we also considered systematic studies of different fuzzy and hybrid fuzzy-based approaches for handling clustering [15]. Data-driven scientific applications can benefit from these frameworks with the ability to trigger scalable computation in response to incoming files processing workloads [16]. In some of the collected research [16] authors introduce an open-source framework to achieve on-premises serverless computing for event-driven data processing applications that features: i) the automated provisioning of an elastic Kubernetes cluster that can grow and shrink, in terms of the number of nodes, on multi-Clouds; ii) the automated deployment of a FaaS framework together with a data storage back-end that triggers events upon file uploads; iii) a service that provides a REST API to orchestrate the creation of such functions; and iv) a graphical user interface that provides a unified entry point to interact with the aforementioned services.

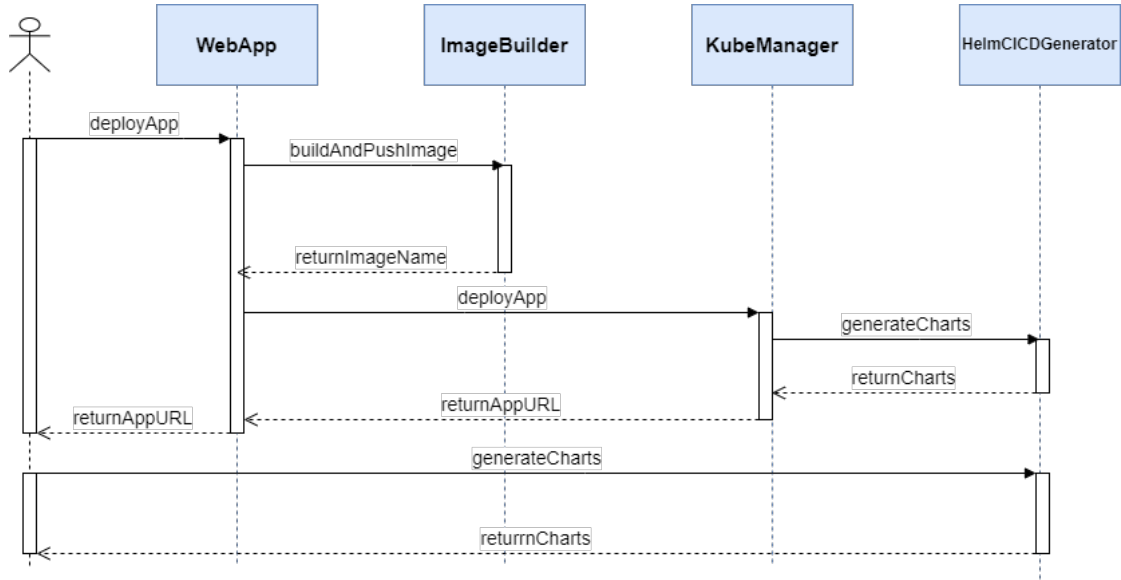
The authors [17] integrate DBSCAN and an Artificial Neural Network capable of automatic route design of ships based on massive AIS data between certain ports. The main objective of [17] is to recognize the key regions by applying the DBSCAN algorithm and then connect these regions automatically by measuring the similarity of the cluster. A fully automatic machine learning platform is designed, which manages server resources uniformly, and users describe the required resources through configuration files. Authors in [18] propose a Cloud infrastructure for the automatic deployment of applications using the services of Kubernetes, Docker, Ansible, and Slurm. Based on multiple node types with a preloaded image (VM or Docker) [19] propose an improved automatic scaling scheme that combines the advantages of different types of nodes in the scaling process to conduct experiments with an actual cluster on the IBM Cloud platform [20, 21]. They are trying to use a delivery cloud platform based on microservices with dedicated services for autonomous driving in order to deploy information to the delivery robot. The authors of the study [22] describe the design and implementation of an ML-based detection system of anomalous pods in a Kubernetes cluster by monitoring Linux kernel system calls (syscalls). Some of the solutions are proposed using a framework based on Markov Decision Process (MDP) and Q-learning to automatically generate optimal defense solutions for networked system states [23, 24].

### 3. Results

Following the distribution of the tool among our colleagues, we conducted an external evaluation to assess its performance and effectiveness among experienced and inexperienced students. Through this evaluation process, we arrived at the conclusive finding that the tool functions as intended, successfully facilitating the deployment of user applications to the cluster. Additionally, we observe that the tool enables users to personalize and customize their deployments according to their specific requirements. This conclusion was reached through rigorous testing and feedback from our colleagues who utilized the tool. They reported positive outcomes, highlighting the tool's capability to seamlessly handle the deployment process and enabling

users to make necessary adjustments based on their application’s unique needs.

These findings validate the efficacy of the tool in achieving its primary objectives and reinforce its value as a reliable solution for deploying applications to a cluster. The positive feedback received from our colleagues further strengthens our confidence in the tool’s functionality. It confirms its potential as a valuable asset for developers and users seeking to streamline and customize their application deployments 1.



**Figure 1:** Sequence diagram of the Initial deployment. Source: author’s contribution.

### 3.1. Implementation Environment

Since the Kubernetes cluster needs to be available and reachable to users at all times, as an implementation environment, we chose a Google Cloud Provider Virtual Machine with Ubuntu 22.04.2 LTS, 4 virtual CPUs, and 16GB of RAM memory. We configured the egress and ingress to allow HTTP and HTTPS communication on ports 22,443,80,8080, and for the user’s applications, we opened ports from 31930 to 32000.

### 3.2. Kubernetes Configuration

We provisioned a Kubernetes cluster of version v1.27.0 through the kubeadm init command. The cluster configuration included utilizing the cri-dockerd network adapter, enabling seamless integration between Kubernetes and the Docker engine. Facilitating external communication with the Kubernetes cluster from beyond the virtual machine (VM), I enhanced an image named dtzar/helm-kubectl:3.11.2. This enhancement involved the integration of the requisite SSH key to facilitate direct retrieval of the Kube config from the VM. This specialized image finds its principal application within CI/CD pipelines, specifically during deployment.

Ensuring the availability of the Kubernetes control plane was paramount during the setup process. This involved ensuring that the control plane could be accessed from outside the virtual machine, allowing GitLab/GitHub runners to utilize Kubectl for access. To achieve this, we needed to set up an Nginx configuration to expose the control plane on port 6443.

Listing 1: Dockerfile for xgolis/deployimage

```
FROM dtzar/helm-kubect1:3.11.2
COPY ./id_rsa ~/.ssh
RUN dos2unix ~/.ssh/id_rsa
```

### 3.2.1. MetalLB

MetalLB is a necessary addition to Kubernetes in bare-metal environments as it lacks a built-in Load Balancer implementation to expose services outside the virtual machine (VM). MetalLB offers a crucial capability in load-balancing Kubernetes services without relying on external hardware. Instead, it leverages the available network resources within the Kubernetes cluster to create a virtual network load balancer, enabling the distribution of traffic to the appropriate pods. This flexibility allows MetalLB to be employed in various environments, including on-premises data centers and public cloud platforms.

By configuring MetalLB version 8.2, services created within the cluster will receive external IP addresses, enabling accessibility outside the VM through a specific port defined in a generated Helm chart. The port value is specified in the Helm chart's values for microservices to obtain an external IP address.yaml file must match the port on which the application runs within the container.

Listing 2: MetalLB yaml configuration. Source: author's contribution

```
---
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: first-pool
  namespace: metallb-system
spec:
  addresses:
  - 10.132.0.0/20
```

## 3.3. Initial Deployment Process

The initial deployment is for users to be able to deploy their application and generate Helm charts and CI/CD pipelines for GitHub and GitLab. The first step involves utilizing a front-end microservice for the initial deployment process. Users are required to provide specific information, such as a Git repository URL, username, application name, Git token, and the port on which the application operates. This information is then used to send a request to an ImageBuilder microservice. We have created four microservices in order to simplify the initial deployment process.

### 3.3.1. WebApp Microservices

The main objective of the Web interface is to allow users to deploy their applications with minimal effort. Once users have filled out the necessary information and clicked the "Deploy" button, the application initiates a request to an ImageBuilder. Once the image is built, another request is sent to a Kubemanager, instructing it to create a namespace using the user's name and deploy the application within it. Subsequently, the web application provides the user with the URL to access their deployed application and allows the user to download GitHub/GitLab CI/CD pipelines for his application. Additionally, the application offers a URL to access the Kubernetes cluster dashboard, which can be reached at [http://<IP\\_OF\\_VM>:32407/](http://<IP_OF_VM>:32407/). This dashboard gives users additional information and controls over their applications deployed within the Kubernetes environment.

### 3.3.2. ImageBuilder Microservices

When a request is made to the ImageBuilder microservice, it responds to several tasks. First, it utilizes the provided Git token to pull the specified Git repository. The microservice then takes the pulled files and creates a tar archive. Next, ImageBuilder builds a Docker image, using the previously created tar archive as the context. This Docker image encapsulates all the dependencies and configurations required to run the application successfully. Once the Docker image is constructed, the ImageBuilder microservice pushes it to the Docker Hub repository. The Docker image is prefixed with *xgolis/* to ensure proper identification and organization. This prefix helps to distinguish and categorize the image within the Docker Hub repository.

### 3.3.3. HelmCICDGenerator Microservice

HelmCICDGenerator microservice generates and modifies Helm charts and CI/CD pipelines for the user's application. Automated CI/CD pipelines are generated by modifying environmental variables in template pipelines. These templates are designed to work seamlessly with both the GitHub and GitLab pipelines. To enable users to utilize these pipelines effectively, they need to insert their docker username (as DOCKERHUB\_USERNAME) and docker token (as DOCKERHUB\_TOKEN) into the respective GitHub or GitLab variables. Generated Github CICD 3 consists of 2 jobs: docker (lines 10-32) and deploy (lines 34-59).

## 3.4. Docker Job

Docker job runs on image ubuntu:latest and is used to build and push an image to a Docker hub. The pipeline performs the following actions:

- **Checkout:** It retrieves a Git repository onto a Git runner, which serves as the environment for executing the pipeline
- **Set up Docker Build:** It configures the Docker engine to utilize Docker Build
- **Login to Docker Hub:** It authenticates with Docker Hub using the provided credentials. Users need to set up these credentials in GitHub Actions variables for the pipeline to log in to Docker Hub successfully

- **Build and push:** It constructs an image using the specified context and proceeds to push it to the user's Docker Hub repository. The image is prefixed with the *user's Docker Hub name*, and *latest* is assigned as the tag.

### 3.5. Deploy Job

The deploy jobs rely on the `xgolis/deployimage 1` image to establish connectivity with a virtual machine (VM) hosting the Kubernetes cluster. These jobs are responsible for deploying a user's application to the cluster.

The steps involved in the deployment process are as follows:

- **Checkout code:** It retrieves the code from a Git repository onto a Git runner, where the pipeline is executed.
- **Deployment:** This step executes a series of bash commands, including:
  1. Line 46: Modifies the permissions of the SSH key.
  2. Line 47: Adds the VM to the list of known hosts.
  3. Line 48: Creates a `.kube` directory.
  4. Line 49: Downloads the Kube config from the VM, including the necessary certificates.
  5. Line 50: Sets the appropriate cluster IP in the Kube config.
  6. Line 51: Configures the Kube config to skip TLS verification.
  7. Line 52: Displays the deployed pods within the user's namespace.
  8. Lines 54-59: Utilizes helm upgrade to apply all the Helm charts to the cluster while specifying values from the `values.yaml` chart.

These steps collectively ensure the successful deployment of the user's application to the Kubernetes cluster.

Listing 3: Github CI/CD pipeline generated by HelmCICDGenerator service. Source: author's contribution

```

1 name: CI
2
3 env:
4   NAMESPACE: exampleUser
5   APPNAME: exampleApp
6
7 on: [push]
8
9 jobs:
10  docker:
11    runs-on: ubuntu-latest
12    steps:
13      -
14        name: Checkout
15        uses: actions/checkout@v3
16      -

```



```

17     name: Set up Docker Buildx
18     uses: docker/setup-buildx-action@v2
19   -
20     name: Login to Docker Hub
21     uses: docker/login-action@v2
22       # ADD YOUR DOCKER USERNAME AND DOCKER TOKEN INTO GITHUB VARIABLES
23     with:
24       username: ${ secrets.DOCKERHUB_USERNAME }
25       password: ${ secrets.DOCKERHUB_TOKEN }
26   -
27     name: Build and push
28     uses: docker/build-push-action@v4
29     with:
30       context: .
31       push: true
32       tags: ${ secrets.DOCKERHUB_USERNAME }/${ env.APPNAME }
33
34   deploy:
35     runs-on: ubuntu-22.04
36     needs: docker
37     container:
38       image: xgolis/deployimage:latest
39     steps:
40     -
41       name: Checkout code
42       uses: actions/checkout@v2
43     -
44       name: Deployment
45       run: |
46         chmod 400 /root/.ssh/id_rsa
47         ssh-keyscan <IP\_OF\_VM> >> /root/.ssh/known_hosts
48         mkdir ~/.kube
49         scp -o StrictHostKeyChecking=no -i /root/.ssh/id_rsa xgolis@<IP\_OF\_
50           _VM>:/home/xgolis/.kube/config ~/.kube
51         kubectl config set-cluster kubernetes --server=https://<IP\_OF\_VM
52           >:6443
53         kubectl config set-cluster kubernetes --insecure-skip-tls-verify
54         kubectl get pods --namespace=${ env.NAMESPACE }
55
56         helm upgrade ${ env.APPNAME } ./${ env.APPNAME } \
57         --install -n ${ env.NAMESPACE } \
58         --set "image.fullImage=${ secrets.DOCKERHUB_USERNAME }/${ env.
59           APPNAME }:latest" \
60         --set "app.namespace=${ env.NAMESPACE }" \
61         --set "app.examplePass=${ env.examplePass }" \
62         --set "app.examplePassword=pass"

```

### 3.5.1. KubeManger Microservices

Once ImageBuilder generates and publishes an image to the Docker Hub, WebApp triggers a request to the KubeManager microservice. The role of KubeManager is to guarantee a successful deployment by first checking if a namespace with the user's name exists. If it does not, KubeManager creates the namespace. Then, it obtains a Helm chart from HelmCICDGenerator and utilizes it to generate a deployment and service. This results in the creation of a pod with an external IP.

## 4. Conclusions

In summary, we have explored the domain of automatic deployment to Kubernetes clusters by using an innovative learning tool and applying novel learning processes. Valuable insights were obtained through a thorough examination of container orchestration technology and an exploration of various automatic deployment principles. The study effectively identified crucial keywords, facilitating an extensive literature search and ensuring a comprehensive grasp of the subject matter. The research findings shed light on the potential of harnessing advanced learning techniques to streamline and optimize the deployment process within Kubernetes clusters. This investigation significantly contributes to the field by offering valuable recommendations and guidelines for practitioners aiming to enhance their deployment practices in Kubernetes environments. By integrating the new learning tool and learning processes, organizations can reap the benefits of improved efficiency, reliability, and scalability in their deployment workflows, ultimately leading to enhanced software delivery and operational excellence.

## Acknowledgments

The work reported here received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 101004887 (rurAllure) and from the Operational Programme Integrated Infrastructure for the project: Support of Research Activities of Excellence Laboratories STU in Bratislava (ITMS code: 313021BXZ1), co-funded by the European Regional Development Fund (ERDF).

## References

- [1] P. Dakić, M. Živković, An overview of the challenges for developing software within the field of autonomous vehicles, in: 7th Conference on the Engineering of Computer Based Systems, ECBS 2021, Association for Computing Machinery, New York, NY, USA, 2021. URL: <https://doi.org/10.1145/3459960.3459972>. doi:10.1145/3459960.3459972.
- [2] L. Benova, L. Hudec, Detecting anomalous user behavior from NGINX web server logs, in: 2022 IEEE Zooming Innovation in Consumer Technologies Conference (ZINC), IEEE, 2022. doi:10.1109/zinc55034.2022.9840589.
- [3] I. Stupavský, P. Dakić, V. Vranić, The impact of fake news on traveling and antisocial

- behavior in online communities: Overview, *Applied Sciences* 13 (2023) 11719. doi:10.3390/app132111719.
- [4] I. Stupavský, P. Dakić, *Antisocial Behavior and the Dopamine Loop on Different Technological Platforms and Industries: An Overview*, Springer Nature Singapore, 2023, pp. 471–481. doi:10.1007/978-981-99-3236-8\_37.
- [5] P. Dakić, V. Todorović, B. Petrović, Investment reasons for using standards compliance in autonomous vehicles, *ESD Conference, Belgrade 75th International Scientific Conference on Economic and Social Development Development, ESD Conference Belgrade, 02-03 December, 2021 MB University, Teodora Dražžera 27, 11000 Belgrade, Serbia; (2021)*. URL: <https://www.shorturl.at/diMRS>.
- [6] L. Benova, L. Hudec, Web server load prediction and anomaly detection from hypertext transfer protocol logs, *International Journal of Electrical and Computer Engineering (IJECE)* 13 (2023) 5165. doi:10.11591/ijece.v13i5.pp5165-5178.
- [7] P. Dakić, V. Todorović, V. Vranić, Financial sustainability of automotive software compliance and industry quality standards, in: *Proceedings of Eighth International Congress on Information and Communication Technology*, Springer Nature Singapore, 2023, pp. 477–487. doi:10.1007/978-981-99-3091-3\_39.
- [8] N. Hroncova, P. Dakic, Research study on the use of CI/CD among slovak students, in: *2022 12th International Conference on Advanced Computer Information Technologies (ACIT), IEEE, 2022*. doi:10.1109/acit54803.2022.9913113.
- [9] T. Golis, P. Dakić, V. Vranić, Creating microservices and using infrastructure as code within the CI/CD for dynamic container creation, in: *2022 IEEE 16th International Scientific Conference on Informatics (Informatics), IEEE, 2022*. doi:10.1109/informatics57926.2022.10083442.
- [10] P. Dakić, V. Todorović, V. Vranić, Financial justification for using ci/cd and code analysis for software quality improvement in the automotive industry, in: *2022 IEEE Zooming Innovation in Consumer Technologies Conference (ZINC), 2022*, pp. 149–154. doi:10.1109/ZINC55034.2022.9840702.
- [11] V. Todorović, P. Dakić, M. Aleksić, Company management using managerial dashboards and analytical software, *ESD Conference, Belgrade 75th International Scientific Conference on Economic and Social Development Development, ESD Conference Belgrade, 02-03 December, 2021 MB University, Teodora Dražžera 27, 11000 Belgrade, Serbia; (2021)*. URL: <https://shorturl.at/diMRS>.
- [12] F. Chalás, I. Stupavský, V. Vranić, Discussion manipulation, language and domain dependent models: An overview, in: *2023 Zooming Innovation in Consumer Technologies Conference (ZINC), IEEE, 2023*. doi:10.1109/zinc58345.2023.10174128.
- [13] S. Venticinque, A. Amato, A methodology for deployment of IoT application in fog, *Journal of Ambient Intelligence and Humanized Computing* 10 (2018) 1955–1976. doi:10.1007/s12652-018-0785-4.
- [14] M. R. Zafar, N. M. Khan, Dlime: A deterministic local interpretable model-agnostic explanations approach for computer-aided diagnosis systems, 2019. doi:10.48550/ARXIV.1906.10263.
- [15] R. Sharma, V. Vashisht, U. Singh, Node clustering in wireless sensor networks using fuzzy logic: Survey, in: *2018 International Conference on System Modeling & Advancement in*

- Research Trends (SMART), IEEE, 2018. doi:10.1109/sysmart.2018.8746977.
- [16] A. Perez, S. Risco, D. M. Naranjo, M. Caballer, G. Molto, On-premises serverless computing for event-driven data processing applications, in: 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), IEEE, 2019. doi:10.1109/cloud.2019.00073.
- [17] Y. Wen, Z. Sui, C. Zhou, C. Xiao, Q. Chen, D. Han, Y. Zhang, Automatic ship route design between two ports: A data-driven method, *Applied Ocean Research* 96 (2020) 102049. doi:10.1016/j.apor.2019.102049.
- [18] R. Doukha, S. A. Mahmoudi, M. Zbakh, P. Manneback, Deployment of containerized deep learning applications in the cloud, in: 2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech), IEEE, 2020. doi:10.1109/cloudtech49835.2020.9365868.
- [19] M. Wang, D. Zhang, B. Wu, A cluster autoscaler based on multiple node types in kubernetes, in: 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), IEEE, 2020. doi:10.1109/itnec48623.2020.9084706.
- [20] S. Choochootkaew, T. Chiba, S. Trent, M. Amaral, Run wild: Resource management system with generalized modeling for microservices on cloud, in: 2021 IEEE 14th International Conference on Cloud Computing (CLOUD), IEEE, 2021. doi:10.1109/cloud53861.2021.00079.
- [21] Z. Yin, J. Liu, B. Chen, C. Chen, A delivery robot cloud platform based on microservice, *Journal of Robotics* 2021 (2021) 1–10. doi:10.1155/2021/6656912.
- [22] R. R. Karn, P. Kudva, H. Huang, S. Suneja, I. M. Elfadel, Cryptomining detection in container clouds using system calls and explainable machine learning, *IEEE Transactions on Parallel and Distributed Systems* 32 (2021) 674–691. doi:10.1109/tpds.2020.3029088.
- [23] T. Smolen, L. Benova, Comparing autoencoder and isolation forest in network anomaly detection, in: 2023 33rd Conference of Open Innovations Association (FRUCT), IEEE, 2023. doi:10.23919/fruct58615.2023.10143005.
- [24] X. Zhou, S. Y. Enoch, D. S. Kim, Markov decision process for automatic cyber defense, 2022. doi:10.48550/ARXIV.2207.05436.