Framework for a Multipurpose Remotely Accessible Laboratory for Education

Vojdan Kjorveziroski
Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University, Skopje, North Macedonia
Email: vojdan.kjorveziroski@finki.ukim.mk

***Abstract*—Hands-on exercises are an important aspect of students' education and they have a positive effect on their overall knowledge retention rates. With the Covid-19 pandemic on one hand, and the ever-increasing number of new students on the other, in-person computer laboratories are no longer a feasible option for the implementation of such practical lessons. The introduction of remote learning laboratories is one potential solution to both of these problems. However, existing remote laboratory implementations only focus on either a single scientific field, are not versatile in terms of supported infrastructure, or support only limited runtime options for the software to be used as part of the exercises. To overcome these issues, we have devised a remote laboratory framework consisting of 8 fundamental feature requirements which would allow flexible use in different fields of study, while at the same time enabling easy extension with existing and new simulation software. We discuss different implementation routes and report on the progress that we have already made in terms of setting up such a remote laboratory for the needs of our courses. We conclude that while most features can be implemented with existing open source software, challenges arise in terms of providing a self-service portal from which students can on-demand deploy resources.**

***Index Terms*—remote learning, remote laboratory, infrastructure orchestration, virtualization, containerization**

I. Introduction

Hands-on experience is one of the pillars of natural sciences education, including computer science. The introduction of real-world practical examples in the curricula directly impacts the students' knowledge retention rates and has an overall positive impact on the education process as a whole [1]. However, in the majority of cases the design and organization of traditional hands-on exercises is a burdensome process for all parties involved, both for the educators and the students alike, no matter whether they are executed remotely or in-person. On one hand the required resources need to be provisioned and configured, while on the other hand it cannot be assumed that all students have the necessary equipment at their personal disposal for successfully taking part in the devised scenarios at home. An obvious solution to the latter problem would be to simply make available on-premise equipment which would be provided for in-person students' use, but even this might prove challenging. In light of recent events, strict lockdowns during the Covid-19 pandemic largely prevented the use of institutional computer science laboratories. Furthermore, the number of natural sciences students increases every year [2] and universities might have logistical difficulties scaling the available resources to students' demand.

One potential solution which would allow to overcome these problems is the introduction of remotely accessible laboratories where the necessary computing infrastructure can be provisioned, configured, and made available to students either by the system administration department of the university or by the course educators themselves. By leveraging either the private cloud infrastructure of the institution or by utilizing public cloud solutions, students can be provided with effortless remote access capabilities, while at the same time not requiring any high-performance infrastructure locally [3]. In such a scenario, the exercise setup would be completed beforehand, without the need for additional effort by the students, thus allowing them to focus on solving the challenges themselves instead of spending time on environment setup.

Should these remote laboratories be implemented in a scalable and easy to maintain manner, their advantages would be many-fold. Firstly, they would facilitate easy collaboration between students who all have access to the same centralized resources on a scale which is simply not feasible when each student completes the exercises on their own local machine or on-premise with a subset of the group. Furthermore, by following security best practices in terms of isolating the remote laboratory from the rest of the production systems, even potentially disruptive exercises can be organized, such as the showcasing of vulnerable software components or practicing with malicious exploits, concepts important for cybersecurity related courses [4]. Finally, by maintaining a pool of historical exercises, students can on-demand, and in their own time, use the infrastructure to revise important concepts in which they might be interested in.

The aim of this paper is to provide a framework for the implementation of a general purpose, infrastructure agnostic, versatile remotely accessible laboratory that can be utilized as part of various university courses requiring hands-on exercises, while not being strictly limited to the area of computer science. The main contributions of this work are:

- Presentation and discussion regarding the state of the art research in terms of implementing remote access laboratories in an education context.
- Definition of a framework for future remote access laboratories, not limited to a particular scientific field.
- Discussion of potential routes in which remote learning laboratories conformant to the framework can be imple-

The work presented in this paper has received funding from the Faculty of Computer Science and Engineering under the "SCAP" project.

mented.
- Reporting on real-world experience acquired during efforts to implement such a remote learning laboratory in a number of computer science related courses.

The rest of this paper is organized as follows: in Section II we discuss related work to the area of remotely accessible laboratories for education purposes, before moving forward with Section III where we outline the set of requirements that we have devised for the implementation of such a remote laboratory. In Section IV we discuss concrete ways in which the requirements outlined in Section III can be met, while also describing various approaches which we have already used during a number of courses at our faculty. We conclude the paper with Section V where we summarize the presented framework and discuss future planned work.

II. Related Work

The need for remotely accessible laboratories that can augment the education process while allowing students to freely experiment and improve their understanding of novel concepts has already been recognized. Efforts have been made to improve the usability of such laboratories by both academia and industry. In the text that follows we outline the state of the art research related to this topic.

Remote learning laboratories have been traditionally very popular in a computer networking context, providing an easy and cost-effective way of gaining hands-on experience with enterprise level devices. Using different forms of virtualization, coupled with advancements in software defined networking (SDN), network devices such as routers or switches can be run on general purpose hardware, eliminating the need for students to buy expensive equipment, thus allowing them to easily acquire much needed experience using the latest revisions of both the software and hardware [5]. Unfortunately the network simulators that make all of this possible usually have very high system requirements in order to be able to virtualize these complex devices and are difficult to set up for inexperienced users taking their first steps in these fields. To alleviate these problems, centralized infrastructures where users can deploy a number of virtualized network devices for training purposes in an ad-hoc manner have been developed. Álvarez et al. [6] present one such implementation of a remote laboratory based on QEMU/KVM virtual machines that users can leverage as an alternative to commercial network training platforms offered by major vendors. Continuing this trend, Scazzariello et al. [7] describe a more modern architecture for deploying virtualized network devices, utilizing a container orchestrator to manage the containerized network devices and ensure the long term scalability of the platform. By developing a custom container network interface (CNI) plugin [8], [9], the authors have managed to adapt the current limitations of the Kubernetes orchestrator of assigning multiple independent interfaces per containers, thus enabling this new use-case.

Another area of particular interest where remote laboratories can be utilized is in the context of cybersecurity exercises. Cybersecurity courses are faced with the challenge of finding a solution to the ever-present problem of how to securely offer vulnerable environments where students can get hands on experience regarding topics that they have only theoretically learned during the curriculum. Sianipar describes Tele-Lab [10] which leverages virtual machines, and in certain cases containers, to offer vulnerable virtual environments which can then be customized using shell scripts after their deployment.

It should be noted that general purpose remote laboratories not targeted at a particular subject matter have been described in literature as well [11]–[15]. Unfortunately, the majority of them focus exclusively on a given runtime technology such as virtual machines [14], [15] or containers [11], [12], thus limiting the number of possible scenarios and impacting the reusability of already created content by forcing reimplementation in the supported runtime environment.

Apart from academia, industry has also identified the benefits of remote laboratories, with many cloud providers offering virtual training for their own products in a sandbox environment [16], [17]. Even though most of these courses are free and accessible to the wider audience, they focus exclusively on the technology of the given cloud provider. Taking into account the low levels of compatibility between offerings of different cloud vendors, vendor lock-in is a major challenge in this area [18].

In conclusion, while there are existing efforts for implementing remotely accessible laboratories for education, they are focused on hosting only certain types of workloads relevant to a particular subject matter, offer limited support for different runtime environments, or can be deployed only on certain types of infrastructure. In our opinion a development of a general purpose remote laboratory framework is needed. Such a framework should be infrastructure agnostic, highly scalable, with support for different runtime environments allowing the deployment of both new and existing workloads. A later practical implementation of the framework would be beneficial not only for computer science education, but also for other scientific disciplines.

III. Requirements Evaluation for a Remote Learning Laboratory

The design of a multipurpose remotely accessible laboratory for education demands careful consideration and evaluation of requirements. Different use-case scenarios might require distinct, often conflicting features. Careful balance must be stricken in terms of supported scenarios, environment isolation, and ease-of-use both in terms of administration and regular usage. On one hand, use-cases such as remote development in various programming languages require a large catalog, supporting different tools and programming languages. On the other hand, strong security measures combined with strict network isolation are essential for scenarios related to cybersecurity, guaranteeing the integrity of the underlying platform itself. Finally, various other fields already have a sizable catalog of simulation software which can only run in certain environments, for example on a given version of an operating system, or only within containers. Effort should be made to

support different runtime environments as to facilitate easy incorporation of existing software into the remote laboratory.

To tackle these challenges, in the remainder of this section we outline 8 fundamental features (abbreviated as F1-F8) for the implementation of a remote laboratory.

- F1: Infrastructure agnostic – The remote laboratory should support deployment both in a virtualized environment, but also directly on bare-metal servers. Different deployment options should be interoperable with each other, allowing the virtual laboratory to be hosted both on physical and virtual servers at the same time. This would ensure both the performance and elasticity of the underlying platform. In times of high resource demand, virtualized resources either from on-premise institutional private clouds or public clouds can augment the permanent hardware resources where the platform itself is hosted.
- F2: Support for different runtime environments – Effort should be made to support both containers and virtual machines as runtime environments for the deployed scenarios and software on top of the platform. This would ensure greater compatibility with existing software that can currently be deployed in a standalone fashion either as containers or inside virtual machines, while saving time which would otherwise be dedicated to conversion. The support for different runtime environments would also future proof the platform, enabling novel technologies such as serverless functions to be utilized, should the need arise.
- F3: Free from vendor lock-in – Even though both virtualization and containerization are general concepts, unfortunately in practice their various implementations are not compatible with each other. This problem is even more pronounced when it comes to hypervisors, with different virtual hard disk formats and guest drivers required for interacting with the host operating system. In terms of containerization, the Open Container Initiative (OCI) [19] aims to create industry standards around container formats and runtimes, allowing container image reuse among different containerization solutions. Nevertheless, the definition of an abstraction layer which will support various virtual machine templates regardless of the hypervisor where they were created is paramount to enable easy migration of existing software solutions to the remote laboratory.
- F4: Integration with existing external systems – The remote laboratory cannot be seen as an isolated silo in terms of the existing and well-proven e-learning systems in use today. The majority of universities have already deployed learning management systems (LMS) for keeping track of students' records and for distributing learning materials. Any platform which would enable the implementation of a remote laboratory needs to support integration with such existing systems, aiding both the registration process of the new users, as well as syncing course participation information necessary for filtering the available scenarios. Furthermore, such an integration should be versatile as to allow both manual and automatic data exchange between the systems which can later also be analyzed to determine students' performance [14].
- F5: End-to-end connectivity between the students and the remote laboratory – Secure end-to-end connectivity between the remote laboratory and the students' computing devices would allow direct communication, no matter the communication protocol that the target software uses. Additionally, such an end-to-end connectivity, might also enable the incorporation of local resources available to the students within the context of an elaborate scenario, a strategy already discussed in [10].
- F6: Efficient use of compute resources – The remote laboratory should offer versatile options for supporting an influx of students. During the implementation process, scenarios where the requested computing capacity exceeds the available capacity should be taken into account. This should not be seen as an unlikely occurrence, since the number of students increases every year, and courses with large number of students which would incorporate the use of the remote laboratory within their curriculum might outpace the rate at which new computing capacity is made available. Multiplexing strategies need to be implemented to deal with such scenarios.
- F7: Effortless collaboration between students – One of the major benefits of a centralized remotely accessible laboratory is that it enables easy collaboration between students no matter their group size. To facilitate this, despite supporting an isolated tenant model, where each student is given an independent remote working environment, shared access to a centralized instance should be possible as well.
- F8: Versatile interfacing options – To avoid requiring any prerequisite knowledge, students and educators alike should be able to choose the interface option most suited to them, either a graphical user interface (GUI) or a command line interface (CLI). The support of a CLI option would make bulk resource provisioning by educators easier, together with requiring less effort for integration with third party systems. On the other hand, a GUI would be necessary from the students perspective, providing easy way to browse the list of supported software and scenarios.

In the section that follows we proceed by outlining guidelines for meeting the stated feature requirements defined in the framework.

IV. Discussion of Implementation Options

The main pillar which would allow the development of a remote laboratory that satisfies all of the previously outlined features is the introduction of a layered design using multiple abstractions. The implementation of such abstraction layers would ensure the wide-ranging compatibility with different virtualization and containerization platforms while providing a

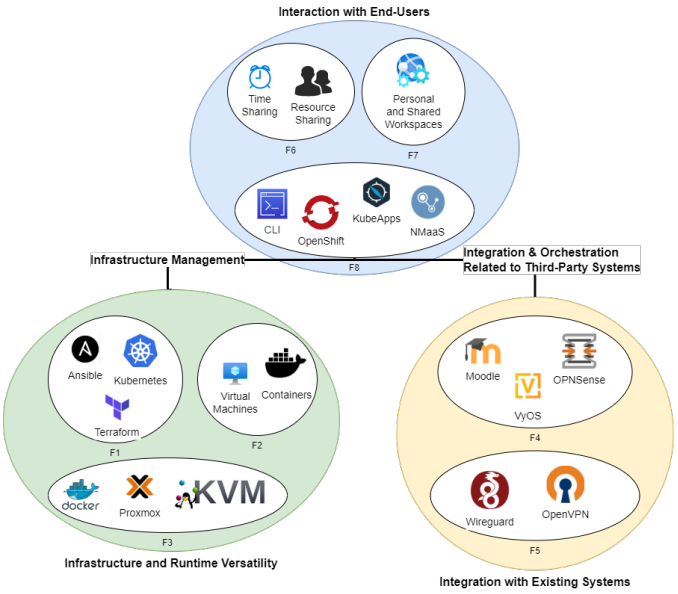

Fig. 1. Visual Representation of the Implementation Options for the Reference Framework

unified view towards the end-users. Fortunately, the advancements made in terms of various configuration management, orchestration, and provisioning tools allow easy integration with diverse systems while relying on a single interface – that exposed by the chosen tool.

Figure 1 provides a visual overview of the implementation possibilities of the framework, outlining the relationship between the various feature requirements. In the remainder of this section, we proceed with the discussion of implementation options for the relevant features, grouped in different subsections.

A. Infrastructure and Runtime Versatility

The F1, F2, and F3 features can be satisfied with an elaborate use of popular and open source provisioning, orchestration, and configuration management tools.

Provisioning tools such as Terraform [20] can be used to deploy infrastructure such as virtual machines, firewalls, routing rules, and containers across different environments, as long as they expose a documented application programming interface (API). With the help of the wider open source community, such tools have implemented compatible modules for various solutions [21], allowing effortless deployment using a unified syntax no matter the underlying platform, thus solving both F1 and F3. With the promotion of the infrastructure as code (IaC) concept [22], infrastructure configuration can be written once and applied across different environments that are compatible with a given simulation software, without the need for any new tooling.

Once the infrastructure has been deployed using a provisioning tool it can be further customized with the help of configuration management solutions, such as Puppet [24] or Ansible [23]. In this manner, different lab scenarios can be described exactly once using the chosen configuration management tool and later shared between users with the aim of extending the number of supported scenarios. Such an approach would allow the creation of scenario catalogs based on IaC scripts written in different configuration management languages. This strategy of describing exercises using configuration management tools has been successfully tested and integrated in practice by the SecGen project [25], [26].

After provisioning and configuration of both the infrastructure and simulation software, the question of its management arises. For this task, orchestration tools can be used which aim to reconcile the actual state of the services with the recorded desired state specified by the administrator. While the use of container orchestrators for managing large fleets of containers has been extensively studied and implemented in practice even in remote learning laboratories [7], their shortcoming is that they are focused solely on a single runtime environment – containers. To satisfy F2, an orchestrator should support both containers and virtual machines at the same time, using the same interface. One possible solution to this problem is the Kubevirt project [27] which modifies the Kubernetes orchestrator to support QEMU/KVM based virtual machines as well. In this manner the Kubernetes scheduler can be used to schedule virtual machines across all cluster nodes, together with containers using a single seamless API, thus fully satisfying F2.

We have already evaluated the use of the SecGen tool to generate randomized virtual machines in the context of a network security course [28]. We have also employed the combination of Terraform and Kubevirt in a more recent instance of the course, deploying both containers and virtual machines on a large scale, for every student taking part. We can report that with the help of these tools it is possible to implement a remote learning laboratory, but without any user-facing interface, students are not able to deploy resources on-demand, and instead the grunt of the work in terms of instance deployment is left to the educators themselves. Options which would allow direct end-user control are discussed in the subsection Interaction with End-Users below.

B. Integration with Existing Systems

To satisfy the requirements set out by F4 and F5 any remote learning laboratory will need to support integration and data exchange with existing systems which are already in use. Unfortunately, in practice, this might be a challenging task, since any third-party component would need to have a well-defined API through which communication can be established. Furthermore, in certain cases different people might be responsible for the maintenance of the remote laboratory on one hand, and the maintenance of the third party systems such as the LMS on another, a fact that would further complicate the testing and enablement of such integrations.

Perhaps the most important and common integration that will be required is that with an existing LMS. Fortunately, many popular learning management systems today expose an API which can be consumed for harvesting relevant student data for the remote laboratory, such as basic information,

enrolled courses, and student performance [29]. In cases where such an API is not available, administrators will need to resort to programmatically executing web requests which would achieve the desired functions by simulating a real client. In our case, we have successfully used both options before; we have utilized the Moodle web service functionality as well as created Python wrappers for programmatically simulating requests, which allowed us to perform actions which are not natively supported by the web service implementation yet. It should be noted though that any large scale implementation of a remote learning laboratory will have to extend this integration towards the end-users, so that they can independently deploy resources directly from the user interface of their choice.

To facilitate remote connectivity to the environment, users will need to resort to the use of virtual private networks (VPNs). As was the case with the LMS, integration with existing systems will be required as well, since many institutions already have VPN concentrators in place both for staff and students. The only requirement is that granular firewall rules can be configured, targeted at each users of the remote laboratory individually. This would ensure network level isolation between the rest of the infrastructure and the users of the remote laboratory, as well as among the users themselves, ensuring that they can only access resources which have been personally deployed. In cases where the infrastructure is being set up from scratch, many popular open source solutions exist that can act as both a firewall and a VPN concentrator, such as PfSense, OPNSense, or VyOS. Care must be taken to first evaluate the quality of any API before making a selection. In our current remote learning environment we are using OPNSense, but since not all required features have been exposed via the API yet, additional tooling had to be created which programmatically executed web requests, simulating a client browser.

C. Interaction with End-Users

The last three feature requirements, F6, F7, and F8 all deal with the way that end-users interact with the remote learning laboratory. A common issue being faced is that the number of students attending a given course drastically varies per year, and as such it might not always be feasible to provision additional hardware resources to meet the new requirements in time. This problem can be solved by introducing some sort of a sharing concept in terms of the compute infrastructure. Perhaps the least intrusive solution for the lack of resources would be to implement a time sharing mechanism, where students can schedule when they would like to use the remote learning laboratory. This would limit the number of users at any single point in time, allowing the infrastructure to meet the demand of users.

Certain scenarios and exercises require team work and the remote learning laboratory together with the implemented security and isolation features should be flexible enough to support this. This can easily be solved with the introduction of the workspaces concept, where each user is given their own dedicated workspace, but should the need arise these workspaces can be shared between multiple users either perpetually or for a limited time. In this manner, resources deployed by a single student in a given resource would be accessible by all other teammates.

The last, and perhaps the most challenging feature to implement is a user facing application that would allow students to access, configure, and deploy resources within the virtual environment by themselves. Such an application should incorporate support for all of the features described previously, such as: allow users to deploy workloads from the catalog depending on the courses that they are enrolled in; allow users to schedule a desired timeslot for using the remote learning laboratory; dynamically generate firewall rules to reflect different sharing states of the workspaces; facilitate easy addition of new exercises by educators and administrators; interface with the different orchestration, configuration, and provisioning tools to support the on-demand deployment of applications within the catalog. To the best of our knowledge, no such tool exists that can satisfy all of these requirements by itself at present. However, the use of Kubevirt as a Kubernetes extension that allows the Kubernetes API to be used to manage virtual machines along with containers would allow the repurposing of Kubernetes focused applications that allow users to select and deploy workloads. Examples include NMaaS [30], OpenShift [31], and KubeApps [32]. However, it should be noted that none of these satisfy all of the features in their current state, so they can only serve as a starting point from where additional modifications will need to be made to their source code.

To solve this final piece of the puzzle, we plan to evaluate the various existing open source solutions and determine whether they can be modified to support all 8 feature requirements or whether a new platform will have to be created from scratch.

V. Conclusion

We have presented a framework consisting of 8 distinct feature requirements for the implementation of a universal and versatile remote learning laboratory, supporting both new and existing simulation software. Through the discussion of possible implementation routes, we conclude that the majority of the requirements can be satisfied using open source software, thanks to the recent advancements made in terms of the robustness of configuration management, orchestration, and provisioning tools. We have also reported on our current progress in terms of implementing a conformant remote learning laboratory for use at our faculty during various courses. Even though this remote laboratory, in its current state, is still work in progress, students have already expressed satisfaction and they prefer assignments which make use of the centralized infrastructure instead of those that must be completed locally. This is understandable, since by making use of the remote laboratory students do not need to configure anything, and can instead focus directly on the task at hand, foregoing any manual setup procedures for the used software. However,

attention should be paid to the level of abstraction introduced when deploying software to be used during the exercises, since too much abstraction would have a negative impact in terms of the students' understanding of the underlying architecture of the used simulation tools. This is especially true for computer science subjects, where manually dealing with the complete installation process can prove beneficial and can be used to teach valuable troubleshooting lessons. Nevertheless, using the presented framework a balanced approach can be implemented, where students are tasked with installing the software manually within their assigned virtual workspaces.

One open issue remains before the conceptual framework can be realized in its entirety. To support the desired level of versatility in terms of runtime environments, supporting both virtual machines and containers, either some of the existing open source platforms will have to be adapted or a new one built from scratch. No matter the route taken, the resulting platform will have to support direct access by students, allowing them to on-demand deploy exercises while providing integration with external systems for acquiring information regarding course enrolment, grades, and dynamically provisioning firewall rules for isolation. We plan to evaluate how the existing open source candidates can be adapted to meet the requirements in a future work, and if the required effort is proven to be infeasible, implement such a platform from scratch, thus concluding all of the 8 outlined features in the presented framework.

REFERENCES

[1] P. M. Stohr-Hunt, 'An analysis of frequency of hands-on experience and science achievement', Journal of Research in Science Teaching, vol. 33, no. 1, pp. 101–109, 1996, doi: https://doi.org/10.1002/(SICI)1098-2736(199601)33:1⟨101::AID-TEA6⟩3.0.CO;2-Z.

[2] 'Tertiary education statistics', Eurostat. https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Tertiary_education_statistics (accessed Apr. 03, 2022).

[3] N. Petrovic, V. Nejkovic, and M. Tosic, 'Dealing with Scalability of Laboratory Sessions in Computer Science University Courses', in 2018 26th Telecommunications Forum (TELFOR), Nov. 2018, pp. 1–4. doi: https://doi.org/10.1109/TELFOR.2018.8612090.

[4] D. C. Rowe, B. M. Lunt, and J. J. Ekstrom, 'The role of cyber-security in information technology education', in Proceedings of the 2011 conference on Information technology education, New York, NY, USA, Oct. 2011, pp. 113–122. doi: https://doi.org/10.1145/2047594.2047628.

[5] R. K. CV and H. Goyal, 'IPv4 to IPv6 Migration and Performance Analysis using GNS3 and Wireshark', in 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), Mar. 2019, pp. 1–6. doi: https://doi.org/10.1109/ViTECoN.2019.8899746.

[6] C. Álvarez, 'Developing a Flexible Virtual Networking Laboratory Platform for Education', in Proceedings of the IV School on Systems and Networks, SSN 2018, Valdivia, Chile, October 29-31, 2018, 2018, vol. 2178, pp. 35–38. [Online]. Available: http://ceur-ws.org/Vol-2178/SSN2018_paper_9.pdf.

[7] M. Scazzariello, L. Ariemma, G. Di Battista, and M. Patrignani, 'Megalos: A Scalable Architecture for the Virtualization of Large Network Scenarios', Future Internet, vol. 13, no. 9, Art. no. 9, Sep. 2021, doi: https://doi.org/10.3390/fi13090227.

[8] R. Kumar and M. C. Trivedi, 'Networking Analysis and Performance Comparison of Kubernetes CNI Plugins', in Advances in Computer, Communication and Computational Sciences, Singapore, 2021, pp. 99–109. doi: https://doi.org/10.1007/978-981-15-4409-5_9.

[9] Megalos CNI. Katharà, 2022. Accessed: Apr. 03, 2022. [Online]. Available: https://github.com/KatharaFramework/Megalos-CNI.

[10] J. H. Sianipar, 'Towards scalable and secure virtual laboratory for cybersecurity e-learning', Universität Potsdam, 2020. doi: https://doi.org/10.25932/publishup-50279.

[11] G. Chen, 'CvLabs: A Container Based Interactive Virtual Lab for IT Education', Georgia Institute of Technology, Technical Report, Apr. 2020. Accessed: Mar. 29, 2022. [Online]. Available: https://smartech.gatech.edu/handle/1853/64898.

[12] L. Tobarra, A. Robles-Gómez, R. Pastor, R. Hernández, A. Duque, and J. Cano, 'Students' Acceptance and Tracking of a New Container-Based Virtual Laboratory', Applied Sciences, vol. 10, no. 3, Art. no. 3, Jan. 2020, doi: https://doi.org/10.3390/app10031091.

[13] T. Budai and M. Kuczmann, 'Towards a Modern, Integrated Virtual Laboratory System', Acta Polytechnica Hungarica, vol. 15, no. 3, p. 14, 2018.

[14] Y. Deng, D. Lu, C.-J. Chung, D. Huang, and Z. Zeng, 'Personalized Learning in a Virtual Hands-on Lab Platform for Computer Science Education', in 2018 IEEE Frontiers in Education Conference (FIE), Oct. 2018, pp. 1–8. doi: https://doi.org/10.1109/FIE.2018.8659291.

[15] M. Perales, L. Pedraza, and P. Moreno-Ger, 'Work-In-Progress: Improving Online Higher Education with Virtual and Remote Labs', in 2019 IEEE Global Engineering Education Conference (EDUCON), Apr. 2019, pp. 1136–1139. doi: https://doi.org/10.1109/EDUCON.2019.8725272.

[16] 'AWS Educate', Amazon Web Services, Inc. https://aws.amazon.com/education/awseducate/ (accessed Mar. 29, 2022).

[17] 'How to get started in cloud computing — Google Cloud Training', Google Cloud. https://cloud.google.com/training/getstarted (accessed Apr. 03, 2022).

[18] M. Armbrust et al., 'Above the Clouds: A Berkeley View of Cloud Computing', undefined, 2009, Accessed: Apr. 03, 2022. [Online]. Available: https://www.semanticscholar.org/paper/Above-the-Clouds%3A-A-Berkeley-View-of-Cloud-Armbrust-Fox/1f7190fc294246f83f1f331cc51e3264851d0d36

[19] 'Open Container Initiative - Open Container Initiative'. https://opencontainers.org/ (accessed Apr. 03, 2022).

[20] 'Terraform by HashiCorp', Terraform by HashiCorp. https://www.terraform.io/ (accessed Apr. 03, 2022).

[21] 'Terraform Registry'. https://registry.terraform.io/ (accessed Apr. 03, 2022).

[22] T. A. Limoncelli, 'GitOps: A Path to More Self-service IT: IaC + PR = GitOps', Queue, vol. 16, no. 3, pp. 13–26, Jun. 2018, doi: https://doi.org/10.1145/3236386.3237207.

[23] 'Ansible is Simple IT Automation'. https://www.ansible.com (accessed Apr. 03, 2022).

[24] 'Powerful infrastructure automation and delivery — Puppet'. https://www.puppet.com/ (accessed Apr. 03, 2022).

[25] Z. C. Schreuders, T. Shaw, M. Shan-A-Khuda, G. Ravichandran, J. Keighley, and M. Ordean, 'Security Scenario Generator (SecGen): A Framework for Generating Randomly Vulnerable Rich-scenario VMs for Learning Computer Security and Hosting CTF Events', presented at the 2017 USENIX Workshop on Advances in Security Education (ASE 17), 2017. Accessed: Mar. 05, 2022. [Online]. Available: https://www.usenix.org/conference/ase17/workshop-program/presentation/schreuders

[26] Cliffe, Security Scenario Generator (SecGen). 2022. Accessed: Apr. 03, 2022. [Online]. Available: https://github.com/cliffe/SecGen

[27] 'KubeVirt.io', KubeVirt.io. https://kubevirt.io/ (accessed Jan. 14, 2022).

[28] V. Kjorveziroski, 'Engaging Students with Personalized and Remotely Orchestrated Cybersecurity Training Exercises', in Proceedings of the 18th International Conference for Informatics and Information Technology, May 2021, pp. 48–53. [Online]. Available: http://ciit.finki.ukim.mk/proceedings/ciit-2021-proceedings.pdf.

[29] 'Web service API functions - MoodleDocs'.https://docs.moodle.org/dev/Web_service_API_functions (accessed Apr. 03, 2022).

[30] 'NMaaS Home - Network Management as a Service - GÉANT'. https://www.geant.org/Services/Connectivity_and_network/NMaaS (accessed Jan. 14, 2022).

[31] 'OpenShift Virtualization'. http://content.cloud.redhat.com/learn/topics/virtualization/ (accessed Apr. 03, 2022).

[32] 'KubeApps, deploy your applications in Kubernetes'. https://kubeapps.com/ (accessed Jan. 14, 2022).