Lightweight Symphony: Towards Reducing Computer Science Student Anxiety with Standardized Docker Environments

Kourtnee Fernalld
kfernalld2018@my.fit.edu
L3 Harris Institute for Assured Information
Melbourne, FL, USA

TJ OConnor
toconnor@fit.edu
L3 Harris Institute for Assured Information
Melbourne, FL, USA

Sneha Sudhakaran
ssudhakaran@fit.edu
Florida Institute of Technology
Electrical Engineering and Computer Science
Melbourne, FL, USA

Nasheen Nur
nurn@fit.edu
Florida Institute of Technology
Electrical Engineering and Computer Science
Melbourne, FL, USA

ABSTRACT

During the COVID-19 pandemic, remote learning (RL) transformed the educational landscape for hands-on Computer Science courses. This paradigm shift accelerated the transition from traditional in-person programming labs to decentralized student-provided resources. Even as students returned to in-person learning, many continued to rely on their personal computers rather than embracing university-provided labs. However, this shift to decentralized, heterogeneous environments introduced a variety of information technology and instructional challenges. The recent emergence of lightweight, container-based virtualization presents a unique opportunity to address these challenges by offering standardized environments on decentralized platforms. To investigate this opportunity, we implemented lightweight virtualization for three undergraduate computer science courses with a total enrollment of 188 students. To understand the challenges and successes of implementing these environments, we surveyed 42 students before, during, and after the three courses. Our survey responses identified that 84% of students adopted our standardized environments, with 75% indicating it contributed to their success. We believe that sharing our experience will prove valuable for instructors who wish to explore adopting container-based virtualization to reduce student anxiety in the modern classroom.

CCS CONCEPTS

• **Social and professional topics** → **Model curricula**; **Computing education programs**.

KEYWORDS

cybersecurity education, student anxiety, virtualization

SIGITE '23, October 11–14, 2023, Marietta, GA, USA

ACM ISBN 979-8-4007-0130-6/23/10.
https://doi.org/10.1145/3585059.3611432

ACM Reference Format:
Kourtnee Fernalld, TJ OConnor, Sneha Sudhakaran, and Nasheen Nur. 2023. Lightweight Symphony: Towards Reducing Computer Science Student Anxiety with Standardized Docker Environments. In *The 23rd Annual Conference on Information Technology Education (SIGITE '23), October 11–14, 2023, Marietta, GA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3585059.3611432

1 INTRODUCTION

The pandemic-induced shift to remote learning has demanded new pedagogical approaches as faculty hastily adopted hybrid and online learning models [2, 11]. Our university adopted a hybrid learning model, providing online and in-person classes. Video conference tools and digital learning platforms replaced traditional lectures and labs [26]. Without access to university computer labs, the decentralized learning approach created vastly different digital environments for students. Even as students returned to in-person learning, many continued to rely on the unique decentralized resources they had adopted rather than embracing university-provided labs. This paradigm shift introduced unique challenges requiring instructors and students to debug and troubleshoot different software, operating systems, and hardware. As such, technical issues and troubleshooting can detract from valuable learning time and contribute to students' anxiety levels as they strive to keep up with coursework and achieve their learning objectives. We hypothesize that lightweight virtualization presents a solution to this challenge by delivering stability through a standard environment.

In this paper, we make the following contributions:

(1) We propose and implement a lightweight, container-based, standardized environment for three undergraduate computer science courses, including introductory programming, operating systems, and cybersecurity.
(2) We collected survey responses from 42 students before, during, and after the courses to explore the challenges and successes of deploying lightweight virtualization for hands-on undergraduate computer science education.
(3) To allow other instructors to build on our initial success, we publish our container build scripts, images, and tutorials at https://github.com/FITSEC/docker-images.

Organization: Section 2 provides background information about virtual machine technology and explores opportunities in the educational landscape. Section 3 addresses the tools and programs each course requires, the current method of obtaining these resources, why these methods fall short, and our approach. The experimental setup and data collection methods are described in Section 4. Section 5 discusses the results, the challenges faced and provides insights for future experimentation. Section 6 investigates previous work on lightweight virtualization and containerization in education. Finally, Section 7 presents conclusions and a plan to advance this research in the future.

2 BACKGROUND

The following section introduces the opportunities created by lightweight container-based virtualization over traditional hypervisor-based virtual machines. We then explore how this technology delivers a solution to the modern hybrid learning environment.

2.1 Hypervisor-Based Virtualization

In previous works, instructors have leveraged hypervisor-based technology to deliver standardized course environments [10, 12, 15, 18, 21, 25]. In this approach, isolated virtual machines emulate physical hardware, including processing, storage, and memory. This virtualization allows users to run entire operating systems within a guest host. Traditionally, classroom environments have embraced Type-2 hypervisors (e.g., Vmware, VirtualBox) that exist between the host operating system and the hardware [3, 14]. Despite the benefits, hypervisor-based virtualization does introduce challenges. Hypervisor-based virtualization requires allocating memory and storage to replicate the RAM and hard disks of the guest operating system. For example, Ubuntu recommends reserving 8 GB of memory and 25 GB of storage [20]. These resource constraints complicate running multiple virtual machines simultaneously [12], limiting student experiments and research. Further, these memory and storage constraints present a significant challenge for student adoption due to the negative impact on the performance of the host operating system.

2.2 Container-Based Virtualization

Docker offers a platform for developing, deploying, and running applications using lightweight container-based virtualization [16]. Similar to hypervisor-based virtualization, container-based virtualization can emulate many operating systems and software. The Docker build process constructs images from a Dockerfile configuration that describes the base operating system and the runtime environment, including additional libraries, dependencies, system tools, and settings. Since Docker packages all the code and dependencies into an image, the resulting image can run consistently and reliably in many computing environments. At runtime, Docker containers provide an isolated execution environment for the image. Containerization allows users to build and deploy less resource-intensive applications than hypervisor-based virtualization. Further, users can export images to repositories (e.g., Dockerhub), allowing rapid remote deployment. The combination of these benefits introduces opportunities for deployment in educational environments.

Container-based virtualization provides greater support for redesigning and redistributing classroom environments. Consider the scenario where an instructor must add additional software to an image. Under hypervisor-based virtualization, the instructor must redistribute the entire new image to the students. Under container-based virtualization, students only need to pull the additional layers created by the new software.

2.3 Virtualization Opportunities

The shift to remote learning (RL) during the COVID-19 pandemic accelerated a transition in the educational learning environments that computer-science students use to complete hands-on labs [2, 11, 19, 26]. However, decentralizing student environments complicates instructor mirroring and issue troubleshooting. Before this shift, instructors conducted hands-on labs in university labs to control the learning flow. Additionally, decentralizing student environments presents challenges for hands-on labs with complex software requirements. Anecdotally, our Introductory Programming course required a specific legacy Java version preferred by the instructor. Additionally, our Operating Systems Concepts course required installing the OS161 [9] operating system and several dependencies. Finally, our Cybersecurity course required several specific tools with complex configurations, libraries, and dependencies.

The increasing heterogeneity of architectures and operating systems introduces challenges. Student workstations may run the Windows, Linux, and MacOS Operating Systems on AMD-64 [1] and ARM architectures. For example, ARM-based M1/M2 MacBooks can only perform limited hypervisor-based virtualization. The new M1 and M2 processors use an ARM-based architecture instead of the more commonly used AMD-64 architecture. Since hypervisor-based virtualization virtualizes at the hardware level, ARM-based processors cannot emulate an AMD-64 [24] architecture. VMWare Fusion only supports limited operating systems, including ARM-based Linux and Windows. Further, users cannot deploy ARM-based virtual machines to AMD-64 systems and vice-versa. Docker presents a solution for computer science-related education in the modern hybrid educational landscape as it offers standardization. Modern hybrid classrooms, consisting of in-person and remote students, will likely consist of different operating systems and architectures. For example, one student could be at home virtually attending lessons on an M1 MacBook Air, another in person on their Intel MacBook Pro, and a third in person on a school-provided Dell desktop. Container-based virtualization can accommodate all of these computers with a single cross-platform image. Docker, an operating system-independent technology, supports Windows, MacOS, and Linux on x86-64, ARM, MIPS, PPC64LE, and S390X architectures [4].

3 METHODOLOGY

To address the challenges of decentralized student environments, we implemented container-based virtualization solutions for three undergraduate Computer Science courses with a total enrollment of 188 students. Table 1 provides an overview of each image. We hypothesized that deploying standardized container-based environments could reduce student anxiety by providing a solution to

Table 1: Summary of Docker images that we provided to 188 students over three courses.

Course	Size	Base Image	Software	Docker Pulls
Introductions to Software Dev. 1	2.29 GB	ubuntu:20.04	noVNC, java jdk, jre, text editors	105
Operating Systems Concepts	3.48 GB	ubuntu:20.04	OS161 [9] and dependencies	42
Introduction to Cybersecurity	8.27 GB	kalilinux/kali-rolling	Over 140 Apt Packages	132

the technical and instructional challenges of decentralized learning. After consulting with the primary course instructors for each course, we built custom Docker images. Further, we posted them to the Dockerhub repository, making them available remotely for all students.

Fundamentals of Software Development: To understand the impact on first-year students, we implemented the Docker image for our *Fundamentals of Software Development* course. This is a first-year required undergraduate course for both our Computer Science and Software Engineering majors. The course discusses the reading, understanding, and writing of small programs in the Java programming language. In addition to the Docker solution we developed, students could complete hands-on labs using the following methods.

(1) Using dedicated university labs, whose availability is limited by competing courses and students.
(2) Using a shared learning and research server accessible to students, faculty, and staff. This server is accessible via SSH on campus and through a VPN off campus.
(3) Installing all packages, dependencies, and configurations on their personal computer.

To accommodate the course requirements, we implemented a docker image that provided Java JDK, JRE, and code editors. Further, we constructed our image from a base image constructed by [7] that offered a graphical desktop environment. Newer students could connect to the image using a web-based Virtual Network Computing (VNC) session to support ease of adoption. Further, we constructed students how to map a persistent folder on their desktop to transfer files into and out of the Docker environment.

Operating Systems : To examine the impact on third-year undergraduate Computer Science students, we implemented a solution for our *Operating Systems Concepts* class. In this course, our students examine the design and implementation of operating systems using the instructional operating system OS161 [9]. During past iterations of this course, students indicated difficulty setting up the course environment before their first assignment. Additionally, the instructor-provided hypervisor-based virtual machine image does not work on the newer M1 and M2 MacBooks. In addition to the Docker solution we developed, students could complete their hands-on labs using the following methods.

(1) Develop their virtual machine based on instructor-provided build instructions.
(2) Download and configure an instructor-provided VMWare image.

We expected the students enrolled in this course to have a higher computer literacy than introductory students, so our Docker solution did not include a browser-accessed VNC session. Further, a previous student in the course built a Docker container that we used as a base for our image [13]. As with the previous container, we provided instructions about mapping a folder on the local filesystem to the container.

Cybersecurity Elective: Finally, we developed a Docker container for our *Introduction to Cybersecurity* course, an elective that explores a breadth of offensive cybersecurity techniques. While we deployed minimal containers for the previous two courses, the breadth of materials in this course required a unique approach. The course has eight hands-on labs and a semester-long CTF competition. In addition to the Docker solution we developed, students could complete these labs using the following methods.

(1) Using a dedicated cybersecurity lab, with 24-7 access provided by student id cards.
(2) Download and configure an instructor-provided VMWare image.

Due to the breadth of the cybersecurity course material, our cybersecurity course adopted hypervisor-based virtualization for four years before our experiment. Initially, the instructor deployed large VMWare images (over 40 GB) quarterly. Instructors reported a large overhead updating and maintaining these images with new distribution releases. By 2021, our instructors had begun experimenting with replacing VMWare images with Docker containers. By our experiment in the Fall of 2022, Docker had entirely replaced VMware as the virtualization of choice for the Cyber Operations courses. The image created for these courses is larger due to the tools and resources required by all the classes it covers. The initial Docker image version for Introduction to Cybersecurity required 20.5 GB of storage. Using the official Docker documentation on optimizing builds, we helped the instructor decrease the build size to 8.27 GB [5].

4 EXPERIMENT

4.1 Experiment Setup

We ran our experiment for the Fall 2022 16-week semester. During the initial class lessons and labs, we provided 188 students with training on deploying Docker resources, mapping the local file system, connecting to network resources, and basic troubleshooting. We developed a set of three optional surveys for each participating class during the 16-week semester.

4.2 Survey

The surveys contained six multiple choice questions, nine Yes/No questions, two 5-point Likert scale questions, two 10-point Likert scale questions, and twelve open responses questions. Our surveys focused on the Docker environments and student anxiety. We collected anonymous responses to each survey using Google Forms via

Table 2: Discrete question responses.

Question	Response
Before	
What kind of computer are you using for this class	Mac (older version): 11 Mac (M1/M2 version): 9 Windows (personal computer): 28 Windows (school computer): 2 Linux (any flavor): 9 Chromebook/Chrome OS: 0
Have you heard of a virtual machine or a container	Y: 56, N: 3
Have you heard of Docker	Y: 20, N: 39
Have you ever used Docker	Y: 6, N: 53
Which course did you plan to use the Docker environment for (check all that apply)	Fundamentals of Software Development 1: 18 Operating Systems Concepts: 22 Introduction to Cybersecurity: 19
Programming/programming language experience (check one)	0-1 years experience: 12 1-2 years experience: 16 2-3 years experience: 16 3 or more years experience: 15
Are you mentally prepared for the upcoming course	Y: 56, N: 3
How would you rate your overall level of anxiety for the upcoming portion of the course (scale of 1-10)	4.8 +/- 2.4738
Midterm	
Did you set up the Docker environment	Y: 38, N: 7
Which course did you use the Docker environment for (check all that apply)	Fundamentals of Software Development 1: 9 Operating Systems Concepts: 21 Introduction to Cybersecurity: 15
How difficult was it to set up the Docker environment (scale of 1-5)	2.4 +/- 1.1998
Has your anxiety about the course improved	Y: 29, N: 11, No Response: 5
How would you rate your overall level of anxiety for the upcoming portion of the course (scale of 1-10)	4.1 +/- 2.4292
Final	
Do you think that the Docker environment worked well for this course	Y: 33, N: 5, No Response: 4
Which course did you use the Docker environment for (check all that apply)	Fundamentals of Software Development 1: 12 Operating Systems Concepts: 19 Introduction to Cybersecurity: 9 Other: 2
Do you feel like the Docker environment helped you in this course	Y: 30, N: 10, No Response: 2
What the anxiety you felt for the course helped at all by the preconfigured and standardized Docker environment	Y: 27, N: 12 , No Response: 3

an opt-in process. We distributed the initial survey during the first week of the semester. It consisted of thirteen questions to establish a student interest, resources, and anxiety baseline. We waited for the stress of midterms to be over before we distributed the second survey. This survey contains ten questions designed to assess how students embraced the Docker environment, their experience so far, and their level of anxiety in the wake of its introduction. We conducted the final survey during the last full week of classes. We asked students eight questions to assess the efficacy of the Docker environment and their anxiety after the course. Table3 provides a summary of the questions.

4.3 Results

Over 22% of enrolled students completed the surveys, including 59 initial responses, 45 mid-course responses, and 42 end-of-course responses. Table 2 records the multiple-choice, yes/no, and Likert-scale questions. Figure 1 depicts a word cloud of an open-ended opinion-based question. Students noted the ease of use, specifically regarding the setup of the container. Additionally, the container

Figure 1: Word cloud of responses from "What have you liked most about the Docker container so far?"

was praised for the convenience it gave users by offering all the tools needed for a given course.

5 LESSONS LEARNED

5.1 Successes

Reported Reduction in Anxiety: All three courses combined showed an overall decrease in anxiety at the midterm and end-of-semester levels. By the end of the semester, 69.2% of students reported that the standardized environment specifically helped reduce their anxiety about their course. 75% of the students reported that they believe the course environment worked helped them, with an even greater 86.8% of reporting the environment worked well for their course in general. Open-ended responses identified the convenience and usability of the Docker images. Table 3 depicts all three courses' initial pre vs. midpoint anxiety. We collected self-reported anxiety levels on a scale of 1 to 10. To classify *high* anxiety, we referenced the Generalized Anxiety Disorder 7 items (GAD-7) which uses a scale of 0-21 [22]. We adjusted this value to our scale of 1-10, and selected 7 as the value for *high* anxiety. For the Introductory Programming course, the high-level anxiety increased from 0% to 33.4%. We believe this could be due to the lack of adoption of the environment or due to the students being new to experiencing University-level coursework. The Operating Systems course originally reported high-anxiety in 24.1% of students before decreasing to 19.1%. Our Cybersecurity class initially reported that 41.2% experienced high anxiety. However, by mid-semester, it was down to 14.2%.

Strategies for Greater Adoption: We hypothesize that minimizing the installation difficulty strongly contributed to successful adoption of our Docker images. We identified and practiced failures deploying our images using the methodology recommended in [19]. We spent four weeks before the semester testing our images on different workstations, operating systems, and architectures to identify potential issues. Students praised this cross-architecture ability that returned usability to their M1 and M2 Macbooks. Further, we experimented with the most efficient ways to distribute the images to the students efficiently and effectively. After creating the Dockerfiles for each course, we initially expected to have students follow the instructions to build the environment on their computers. During one of the earlier rehearsals, we discovered building the image from the Dockerfile across multiple platforms occasionally presented challenges. For example, the image on the Windows test machine did not build correctly or function. M1 Macbooks would build ARM-based Linux distributions instead of the required AMD64 architecture. In response, we began self-building images and pushing the images to the DockerHub image hosting registry. For $60 annually, Dockerhub supports 5,000 daily image pulls, five concurrent builds, and 300 vulnerability scans [6]. We built the images for each course on an AMD64/Linux machine and uploaded the compiled images to the DockerHub Registry. We then provided students with a simple *one-liner* command to pull the image, map a local folder, start, and attach. We argue that this approach significantly reduced the opportunity for errors in configuration, building, and deploying images.

Greater adoption for higher level courses: We observed greater adoption among our Cybersecurity and Operating Systems courses with 100% adoption rates compared to a 78% adoption in the Software Development course. We hypothesize that students in the higher level courses have a lot more experience with environment setup and troubleshooting. We further argue that these students were more likely to use a preconfigured environment because they understand the time constraints of configuring and installing software. Seven open-ended responses noted that the Cybersecurity course included several preinstalled tools and programs, allowing them to solve several hands-on course homework and labs. Eleven open-ended responses identified the ease of deployment for the Operating Systems container.

Extending to Extracurricular Activities: Our university fields a cybersecurity competition team. Due to the constant student turnover, the team conducts routine training sessions once a week for newer students. We observed that our cybersecurity team adopted the Cybersecurity course container. Anecdotally, our team's leadership reported the ease of deployability introduced a significant opportunity to save time during training sessions. Instead of helping newer students install a Linux-based environment or a specific tool, they would tell them to pull the Cybersecurity container containing all the necessary tools.

5.2 Challenges

Managed Environment Challenges: Our university operates a Windows Domain environment, with university labs containing restricted Windows 11 workstations. Instructors do not have permission to add or configure software to the workstations. Instead, they must submit tickets to the IT help desk. This restriction presented a significant challenge as we tried to extend our Docker containers to our university labs. Our IT encountered technical issues installing Docker and the required Windows Subsystem for Linux (WSL.) Our IT staff's lack of familiarity with container-based virtualization complicated these issues. It took four weeks to deploy Docker and WSL to our classroom environments.

Understanding the Audience: We struggled with the early and overall adoption of the Docker environment with first-year students. Computer Science and Software Engineering students must

Table 3: We surveyed students about their anxiety for the upcoming course on a scale of 1-10 at two points during the semester. Our results show an overall decrease in anxiety from the beginning to the semester midpoint for each class.

Course	Initial Anxiety (Respondents)	Midpoint Anxiety (Respondents)
Introductions to Software Development 1	4.2222 +/- 2.8191	4.3333 +/- 2.958
Operating Systems Concepts	5.8333 +/- 2.4070	4.6428 +/- 2.2397
Introduction to Cybersecurity	5.1724 +/- 2.3914	3.75 +/- 2.4251

take the Fundamentals of Software Development I course as part of their introductory programming requirements. These students often need more experience with computing. We developed and distributed instructions for installing Docker and deploying the course environment. However, the first-year students reported difficulty following the instructions. In response, we rewrote and redistributed the instructions. Unfortunately, we observed that students often needed to re-engage with the updated instructions and adopt the Docker container as their programming environment of choice. In future course offerings, we intend to conduct rehearsals with first-year students to ensure ease of understanding of the installation, configuration, and deployment.

Instructor Buy-In Increases Adoption: Convincing students to adopt new technology voluntarily, even for their benefit, presents challenges. Learning new things can be daunting, and learning a complicated technology can increase that anxiety. A GSA introduced the Docker environment during the lab portion of the Fundamentals of Software Development 1 course. The instructor briefly mentioned it during the lecture but should have emphasized it more. The beginner students that make up this course may choose a more familiar environment over a more beneficial one. Students taking the Introduction to Cybersecurity course saw a mixed adoption rate. The instructor introduced the standardized environment to the class. The instructor encouraged its use, informing students that help was only available for troubleshooting issues within the educational environment. Despite introducing the Docker environment, many students opted to configure their personal Linux environments. We saw the highest adoption rate of students surveyed at 100% in the Operating Systems Concepts course. The instructor for this course heavily pushed the standardized environment to replace the previous methods for environmental setup. Using the Docker environment was particularly helpful to students with the M1 and M2 MacBooks to avoid configuring a VM on an ARM architecture.

6 RELATED WORK

Previous works have explored the use of container-based virtualization to provide students with a standardized environment [10, 12, 15, 21, 25]. Tobarra et al. previously argued for integrating remote virtual labs into the classroom in [25]. The authors used a Docker container-based virtualization to develop remote virtual lab environments. However, this presents a challenge as students would have to rely on remote resources' dependability and access to consistent and reliable internet. This approach presents significant connectivity hurdles. In less accessible environments, providing students with local resources on their computers might be a solution to the problem [10]. In response, we argue for delivering lightweight containers to run on student workstations regardless of connectivity. Our previous work has argued the value of Docker virtualization for a cybersecurity concentration [17–19, 23]. Jiang and Song examined container-based virtualization in classrooms and laboratories in [10]. The paper explains the advantages and disadvantages of container- and hypervisor-based virtualization. The authors address four different courses that experienced difficulties based on working environments. Their investigation focused on leveraging pre-built Docker Hub images only. We argue that Docker's relative ease allows instructors to create custom-made images specifically tailored for their courses. Alternatively, [15] transitioned their campus cluster of Linux systems to the cloud. After the launch of Codespaces on Github in 2021 [8], they began using it to distribute standardized coding environments to students. Codespaces, however, is a cloud-based version of Visual Studio Code backed by Docker containers. While this approach may satisfy introductory programming courses, it does not scale to meet the needs of more complex coursework like operating systems, databases, networking, cybersecurity, or machine learning.

7 CONCLUSION

This paper presented our experiences introducing lightweight virtualization into the classroom. We hypothesized that this approach would reduce student anxiety by delivering stability through a standard environment. We explored this hypothesis by designing and implementing standardized Docker environments for three undergraduate computer science courses. Students reported positively about our approach. Out of 42 Students, 75% reported that Docker contributed to their success during the course. Our initial experiments identify that further research is necessary to understand the benefits of integrating lightweight virtualization in the classroom. Our work underscores the importance of developing adaptable educational environments to reduce anxiety and enhance learning outcomes when applied to decentralized learning. We share our materials and experiences for reproducibility, offering insight for instructors who wish to embrace lightweight virtualization in the classroom.

ACKNOWLEDGMENTS

This material is based upon work supported in whole or in part with funding from the Office of Naval Research (ONR) contract #N00014-21-1-2732. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the ONR and/or any agency or entity of the United States Government.

REFERENCES

[1] Akinlolu Adekotujo, Adedoyin Odumabo, Ademola Adedokun, and Olukayode Aiyeniko. 2020. A Comparative Study of Operating Systems: Case of Windows,

UNIX, Linux, Mac, Android and iOS. *International Journal of Computer Applications* 176 (2020), 16–23.

[2] Sanaa Ashour, Ghaleb A El-Refae, and Eman A Zaitoun. 2021. Post-pandemic higher education: Perspectives from university leaders and educational experts in the United Arab Emirates. *Higher Education for the Future* 8, 2 (2021), 219–238.

[3] Manish Bhatt, Irfan Ahmed, and Zhiqiang Lin. 2018. Using virtual machine introspection for operating systems security education. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, Minneapolis, MN, 396–401.

[4] Docker Inc. 2023. Docker Docs: Multi-platform images. https://docs.docker.com/build/building/multi-platform/. Accessed: May 9, 2023.

[5] Docker Inc. 2023. Optimizing builds with cache management. https://docs.docker.com/build/cache/. Accessed: June 8, 2023.

[6] Docker Inc. 2023. Pricing and Subscriptions. https://www.docker.com/pricing/. Accessed: June 8, 2023.

[7] Frederic Boulanger. 2016. docker-ubuntu-novnc. https://github.com/Frederic-Boulanger-UPS/docker-ubuntu-novnc. Accessed: August 17, 2022.

[8] Github. 2023. Github: Start coding instantly with Codespaces. https://github.com/features/codespaces. Accessed: May 9, 2023.

[9] David A Holland, Ada T Lim, and Margo I Seltzer. 2002. A new instructional operating system. *ACM SIGCSE Bulletin* 34, 1 (2002), 111–115.

[10] Keyuan Jiang and Qunhao Song. 2015. A preliminary investigation of container-based virtualization in information technology education. In *Proceedings of the 16th Annual Conference on Information Technology Education*. ACM, Chicago,IL, 149–152.

[11] Nicole Johnson, George Veletsianos, and Jeff Seaman. 2020. US Faculty and Administrators' Experiences and Approaches in the Early Weeks of the COVID-19 Pandemic. *Online Learning* 24, 2 (2020), 6–21.

[12] Tae-Hoon Kim, Keyuan Jiang, and Vivek Singh Rajput. 2016. Adoption of container-based virtualization in IT education. In *ASEE Annual Conference & Exposition*. ASEE, New Orleans, LA, 1–13.

[13] Louie Orcinolo. 2022. os161 docker. https://github.com/condor0010/os161_docker. Accessed: August 17, 2022.

[14] Dale L Lunsford. 2009. Virtualization technologies in information systems education. *Journal of Information Systems Education* 20, 3 (2009), 339.

[15] David J Malan. 2022. Standardizing Students' Programming Environments with Docker Containers: Using Visual Studio Code in the Cloud with GitHub Codespaces. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 2*. ACM, Dublin, Ireland, 599–600.

[16] Dirk Merkel. 2013. Docker: Lightweight Linux Containers for Consistent Development and Deployment. https://www.docker.com/.

[17] TJ OConnor. 2022. HELO DarkSide: Breaking Free From Katas and Embracing the Adversarial Mindset in Cybersecurity Education. In *Special Interest Group on Computer Science Education (SIGCSE)*. ACM, Providence, RI, 710–716.

[18] TJ OConnor, Carl Mann, Tiffanie Petersen, Isaiah Thomas, and Chris Stricklan. 2022. Toward an Automatic Exploit Generation Competition for an Undergraduate Binary Reverse Engineering Course. In *Innovation and Technology in Computer Science Education (ITiCSE)*. ACM, Dublin, Ireland, 442–448.

[19] TJ OConnor and Chris Stricklan. 2021. Teaching a Hands-On Mobile and Wireless Cybersecurity Course. In *Innovation and Technology in Computer Science Education (ITiCSE)*. ACM, Virtual Event, 296–302.

[20] Oliver Smith. 2021. How to run an Ubuntu Desktop virtual machine using VirtualBox 7. https://discourse.ubuntu.com/t/how-to-run-an-ubuntu-desktop-virtual-machine-using-virtualbox-7/25137. Accessed: June 8, 2023.

[21] Prateek Sharma, Lucas Chaufournier, Prashant Shenoy, and YC Tay. 2016. Containers and virtual machines at scale: A comparative study. In *Proceedings of the 17th international middleware conference*. ACM, Trento, Italy, 1–13.

[22] Robert L Spitzer, Kurt Kroenke, Janet BW Williams, and Bernd Löwe. 2006. A brief measure for assessing generalized anxiety disorder: the GAD-7. *Archives of internal medicine* 166, 10 (2006), 1092–1097.

[23] Chris Stricklan and TJ OConnor. 2021. Towards Binary Diversified Challenges For A Hands-On Reverse Engineering Course. In *Innovation and Technology in Computer Science Education (ITiCSE)*. ACM, Virtual Event, 102–107.

[24] Songpon Teerakanok, Ittipon Rassameeroj, Assadarat Khurat, and Vasaka Visoottiviseth. 2022. Lessons Learned from Penetration Testing Hands-on Training during COVID-19 Pandemic. In *2022 6th International Conference on Information Technology (InCIT)*. IEEE, Lisbon, Portuga, 368–373.

[25] Llanos Tobarra, Antonio Robles-Gomez, Rafael Pastor, Roberto Hernandez, Andres Duque, and Jesus Cano. 2020. Students' acceptance and tracking of a new container-based virtual laboratory. *Applied Sciences* 10, 3 (2020), 1091.

[26] Jordan R Wlodarczyk, Erik M Wolfswinkel, and Joseph N Carey. 2020. Coronavirus 2019 video conferencing: preserving resident education with online meeting platforms. *Plastic and Reconstructive Surgery* 146, 1 (2020), 110e–111e.