2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC

Host Sponsor UNC Charlotte

ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION

Paper ID #51818

An approach for laboratory assignments for a course in cloud computing using desktop computers and widely available software

Prof. Thomas J. Hacker, Purdue Polytechnic Institute, Purdue University – West Lafayette

Thomas J. Hacker is a Professor of Applied and Creative Computing at Purdue University in West Lafayette, Indiana. His research interests include cyberinfrastructure systems and cloud computing.

An approach for laboratory assignments for a course in cloud computing using desktop computers and widely available software

Abstract

This paper describes an approach developed for laboratory assignments for a graduate course in cloud computing that is cost effective, simple, and can use existing institutional computing infrastructure. The effort was motivated by the need for laboratory assignments that did not require the use of commercial cloud services. The paper describes the curriculum and implementation approach used for a sequence of progressive laboratory assignments that was developed for a cloud computing class that were based on widely available software and desktop computing systems.

The laboratory assignments provided experience in defining, creating, and using infrastructure components, as well as experience in creating and using virtual machines and containers. The technologies used for the laboratory assignments described in this paper include Linux, Vagrant, VirtualBox, Docker, Kind, and Kubernetes which all can be run on desktop computing systems in a reproducible manner using Vagrant. The paper includes a reflection on some of the lessons learned from using desktop computers for the laboratory assignments.

The use of existing desktop computers in a computing teaching laboratory demonstrated their utility for teaching cloud computing concepts. The approach described could be useful for others seeking to develop lab assignments for similar courses in cloud computing that can use widely available software, desktop computers, and institutional computing resources.

Introduction

As part of an effort to introduce cloud computing topics within the author's department, a new graduate course in cloud computing was developed during 2024 that was first offered in the Spring 2025 semester. *This paper is focused on and is limited to the first offering of this new course in the Spring 2025 semester.* One objective of the course was to teach students about the infrastructure used for cloud computing and the process and hardware and software components necessary to build a small functional cloud computer system that could be used to run containers. The course sought to provide the education and training needed by students to achieve several learning objectives for students completing the course. These learning objectives included students learning to install, configure, manage, and use the technologies needed to run Kubernetes using Kind (Kubernetes in Docker) [1] [2] on a virtual machine running on a local desktop

computer. The class included a separate scheduled laboratory course section that was also taught by the course instructor. The laboratory assignments developed for this course sought to provide a way for students to practice and ”try out” concepts that were described in class to help supplement the teaching provided in lectures.

These laboratory assignments required access to adequate computer resources and the availability of open source or widely available software packages that students could use for lab assignments. Local computing resources and commercial cloud systems available over the Internet were two potential sources of computing infrastructure. However, commercial cloud systems had some inherent limiting factors that would have complicated their use for the course. Commercial cloud services from providers such as Amazon Web Service and Google Cloud offer programs for requesting access for teaching through the use of credits that can provide a limited amount of resources for use for a class [3] [4]. A brief summary of some of these programs is described on a 2023 Kion.io website [5]. Strain [4] described the authors' experience using AWS for a course. This was a potential option, however, commercial cloud services can be complex to set up, and can require effort during the semester to manage resource use of the commercial cloud computing system (Strain [4] briefly describes the authors' experience with AWS). Other related work is described in the next section of this paper.

Due to the potential costs of using a commercial cloud computing system for the course, and the management overhead necessary to manage the use of allocated cloud computing resources over the semester, this option was not pursued for the course. One goal was to provide students with experience in creating their own cloud computing infrastructure using commodity hardware and widely available software. Learning these concepts independently of commercial systems could help students to learn about core concepts and infrastructure components separately from commercial implementations.

Other factors considered in the decision to use local resources included student experience, and user and resource management. In terms of student experience, providing students with access to a physical computer they could directly work with instead of a more abstract remote server could help to make the infrastructure aspect of the assignments more concrete. Also, one goal was for students to understand the potential transportability of a reproducible Infrastructure-as-Code (IaC) [6] approach. The use of VirtualBox [7] and Vagrant [8] for students' assignments provided a clear path for transportability (being able to use Vagrantfiles for relaunching and configuring the VirtualBox VMs) and infrastructure reproducibility. In terms of user and resource management, the use of local computing resources facilitated clearer in-house control of user accounts, access, and security for students' use of the computing resources. Students would not need to create separate non-institutional accounts for a commercial cloud computing system. Additionally, there was less risk of incurring unexpected additional external costs or exhausting a resource allocation from a commercial cloud computing provider during the semester. Finally, this approach provided a helpful ”sandbox” for lab assignments in which students could let VMs and containers run without the risk of overusing and potentially exhausting a resource allocation from a commercial cloud provider.

An instructional laboratory of Dell OptiPlex Tower Plus 7020 desktop computers, each containing an Intel Core i7-14700 processor with 20 cores, 64 GB of RAM, a 1 TB NVMe drive, and an on-board 1 gigabit Ethernet network interface, were available for cloud computing classes. These

systems provided a local resource that could be used by students for lab assignments. Based on the availability of these desktop computers in a room that could be scheduled, the factor of student experience, and user and resource management, it was decided to use these local computing resources.

The next section of this paper will describe related work, then provide an overview of the class and describe some of the learning objectives that motivated the approach described in this paper. Following this, the paper will provide a summary of the software components and technologies used, and describe the sequence of lab assignments that allowed students to progressively build up a software stack of components to create a local Kubernetes system. Following this, the paper will describe some of the lessons learned and conclusions.

Related Work

Other related work that is focused on the design and use of cloud computing systems that are used for education falls into two broad areas: infrastructure and architectural approaches for designing, implementing, and operating an environment for supporting networking, big data, and cloud related teaching activities (described in Cutting [9], Salib [10], Xue [11], and Neupane [12]); and related work in the use of Docker containers for computing courses (Sedov [13]), cloud platforms for Operating Systems courses (Holovnia [14]), and cloud computing platforms for Cyber ranges for cybersecurity education (Chouliaras [15]).

The first area, cloud education platforms, consists of papers that describe different architectural and implementation approaches used for creating and operating cloud environments or front ends for operating cloud infrastructure. Salib [10] described the use of OpenStack as a private cloud computing platform which used desktop computers in their implementation. This platform was used for migrating laboratory exercises from native Linux or VMware systems to an OpenStack platform deployed by the group. The work described in this paper differs from Salib's paper in two ways. First, Salib focuses on OpenStack. This is in contrast to the approach described in this paper which is focused on the use of the combination of Docker, Vagrant, VirtualBox, and Kubernetes in Docker (Kind). Second, Salib was seeking to migrate lab exercises from prior computing platforms, whereas the effort in this paper is focused on lab exercises for a new class that had not been previously offered.

In another paper, Neupane [12] described the "*Mizzou Cloud DevOps*" learning platform that provided a comprehensive online computing platform that was designed to support students who were learning cloud DevOps approaches and technologies. Neupane's approach is designed to use public clouds (such as AWS and GCP) and existing cloud platforms such as the National Research Platform Nautilus Cloud. Neupane's approach differs from the work described in this paper in that this paper focuses on the efficient use of desktop computer systems to host the computing components along with a progressive approach to build infrastructure on locally managed desktop computers. This is in contrast to Neupane's work that sought to provide a gateway for the use of other operating cloud infrastructure systems.

In their paper, Cutting [9] focused on the design and implementation of a private cloud to create a "…vendor-agnostic private cloud toolchain…" [9] for education that was based on free software. Cutting's approach used Docker, Git, Kubernetes, and Rancher [16], and used a separate

computing cluster for Kubernetes. Cutting's work differs from the effort described in this paper in that the approach in this paper used a single desktop computer that did not rely on the availability of a centrally managed cluster for Kubernetes operations. Cutting's efforts are similar in that the work in this paper and Cutter's approach were motivated by the need for a private cloud focused paradigm that could be operated within an institution that wasn't closely tied to a public cloud computing provider.

In another paper, Xue [11] describes the "*Kube-CC*" platform that was based on Kubernetes and was designed for use for supporting big data courses. *Kube-CC* provides a comprehensive cluster, host, user, and storage management system. Xue's work differs from the work described in this paper, which is focused on a progressive instructional framework for infrastructure definition and instantiation for a cloud computing system on a desktop computer.

The second area of related work includes work on the use of Cyber ranges and Operating Systems education. Sedov [13] provides a summary of related work focused on the use of Docker containers for computing courses. The work described in this paper differs from Sedov's paper in that the lab assignments described in this paper provide a platform for Docker containers (using Vagrant) and build upon Docker containers to provide a functional Kubernetes system for students (using Kind).

Holovnia [14] provides a survey of cloud platforms that could be used for teaching Operating Systems that is focused on the use of open source and free software packages. While Holovnia's paper provides a survey of related work, in contrast this paper is focused on a progressive approach for assignments using a Kubernetes in Docker (Kind) based system based on infrastructure components such as Vagrant and Docker that is structured to support a progressive learning approach for students taking the course.

Finally, Chouliaras [15] describes the design of a platform for a Cyber range that is based on OpenStack and Docker intended for use for cybersecurity education and research. Chouliaras also provides a summary of related work focused on computing infrastructure for cyber range platforms. The work in this paper differs from Chouliaras in that this paper is focused on cloud computing infrastructure rather than the application area of cyber ranges.

Overview of the cloud computing course and course learning objectives

The graduate level course in cloud computing (which upper level undergraduates could also take) provided an advanced introduction to the technology components used for cloud computing systems to support applications that can run in containers. The technologies covered in the course included Linux container systems, virtual machines, Kubernetes [17], Helm [18] [19], and Vagrant [8]. Starting with a basic knowledge of Linux, cloud computing technology, and computer networking, the course sought to provide a progressive sequence of topics that progressively built upon prior knowledge that led up to a running Kubernetes installation using Kind (Kubernetes in Docker) [1]. The overall expected outcome for the course was for students who successfully completed the course to gain the skills and experience needed to install, configure, and use the underlying technology components needed to run Kubernetes and the core Kubernetes components needed to run a small cloud computing system.

Overall, the course sought to provide the education and training necessary for students completing the course to achieve several expected learning objectives. These learning objectives included three broad groups of activities: reproducible approaches and tools for defining and building computing infrastructure components to support Kubernetes; knowledge and skills necessary to install and use Kubernetes for running and managing containers; and monitoring, customizing, and extending Kubernetes through the use of add-on software packages.

The first broad activity area related to the computing infrastructure platform and approach needed to create a Kubernetes installation. This activity area included a learning objective to learn to use an Infrastructure-as-Code (IaC) approach (specifically using Vagrant [8] with a Vagrantfile) for defining and launching computing infrastructure for cloud computing. In his book [6], Morris describes the Infrastructure-as-Code approach in detail. This area was related to the concept of reproducible infrastructure [6] that can be built or rebuilt when necessary for scaling up infrastructure or rebuild infrastructure if the cybersecurity of the system may have been compromised. Students completing the course would learn to install, manage, and use techniques and software tools for a container management system (such as Docker [20]), virtual machines (such as VirtualBox [7]), and IaC tools (such as Vagrant [8]) needed to create a reproducible computing infrastructure upon which Kubernetes can be installed and used.

The second area of activity was directly related to Kubernetes. Since Kubernetes functions as a container orchestration system [2], the first aspect of this activity was focused on the definition, use, and management of containers using the Docker container system. After containers, the next aspect of this activity area was focused on the installation of Kubernetes. The Kind software package [1] (which stands for Kubernetes in Docker) is based on predefined Docker containers that can be used to easily and quickly launch a multi-node Kubernetes installation. Kind was used by students in their lab assignments to quickly bring up a functional basic Kubernetes installation. Kind uses published prebuilt Docker containers that represent Kubernetes nodes when running (each running container represents a Kubernetes node). Within each Kind docker container, Kind uses the *containerd* [21] container management system to run the Kubernetes components needed for a Kubernetes installation. The final aspect of this activity was focused on learning about the detailed components of Kubernetes [17] including the Container Storage Interface, Container Network Interface, and Kubernetes nodes along with the use of these components.

The final broad area of learning objectives was related to tools and applications that can be used to add functionality to a Kubernetes installation (specifically Helm [19]), monitor a Kubernetes installation (using Prometheus and Grafana [22]), and on how to launch containerized applications on a Kubernetes cluster such as NGINX (an example describing the use of Kubernetes with NGINX is described be Vyas in [17]).

Throughout the course, concepts and topics such as infrastructure reproducibility and the security and licensing implications of containers were discussed. The goal of these topics was to help students understand the need for continual awareness of infrastructure security and to understand the provenance and licensing requirements of software that can be launched on a Kubernetes cluster.

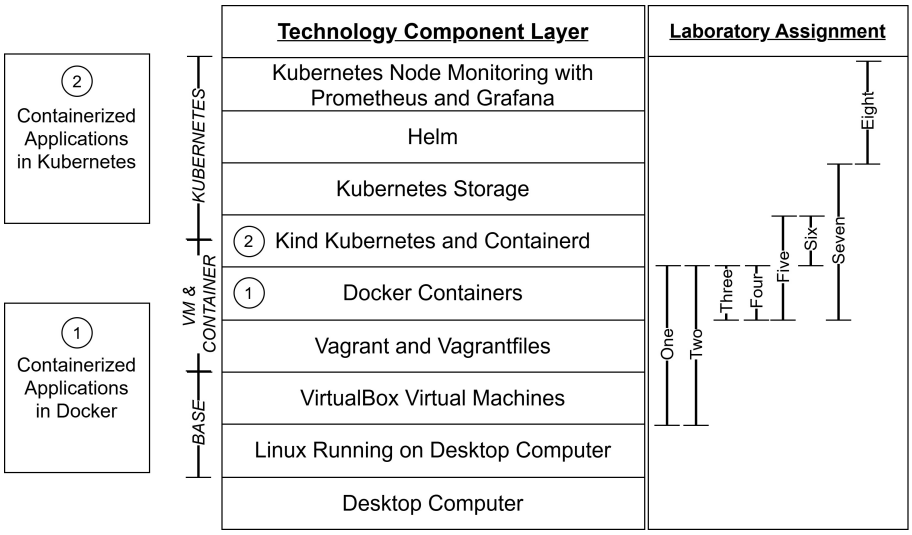


Figure 1: Relationship of technology component layers and laboratory assignments.

Progressive sequence of lab assignments and technology layers

This section provides a summary of the progressive sequence of lab assignments along with a brief overview of the technologies covered in the course (shown in Figure 1). This section also describes how these technologies could be used on a desktop computing platform as part of a cloud computing system for the course. Following this section, the next section will describe each laboratory assignment and the technologies used for each assignment. This paper is limited to the course assignments and syllabus used during the Spring 2025 semester. This study for this paper was reviewed by the Purdue University Institutional Review Board (study number IRB-2025-1311) and was determined to have exempt status.

To describe the relationships among the laboratory assignments and the technologies used for each assignment, this section follows a progressive sequence of layers starting from base technology up to the top technology layer that builds functionality on a running Kind Kubernetes installation. Specific technology areas that were covered in the course originate from the base level of the Linux system running on a desktop computer and goes up through containerized applications that can be run on a cloud computing system based on Kubernetes to the final layer of Kubernetes monitoring.

Figure 1 shows the layers of technology components and the relationships of each laboratory assignment to each technology layer. This section summarizes the technology layers. The next section will describe each laboratory assignment and the technology layers related to each assignment.

The layer *VirtualBox Virtual Machines* that was built on a Linux system used Oracle VirtualBox [7] to create and manage virtual machines. Vagrant [8] can interface with the VirtualBox system to create, load, and delete VMs running within VirtualBox. The *Vagrant and Vagrantfiles* layer used a Vagrant [8] based approach for IaC and system reproducibility. This approach provided a useful platform for teaching students about the necessity for system reproducibility if the desktop computer they used for assignments was rebooted across lab sessions, or if they had a problem

that required the re-creation of their virtual machine. The *Docker Containers* layer was focused on containers and was based on the use of the Docker container system [20]. This level can be used to run programs within a Docker container ("Containerized Applications" - shown as ① in Figure 1) by directly using *docker* commands with the Docker system. As shown in Figure 1, the first four laboratory assignments were focused on these layers. The *Kind Kubernetes and Containerd* layer used Kind [1] to create Kubernetes services that were run within virtual Kubernetes *nodes* (each "node" was run as a Docker container). Each virtual Kubernetes node used the *containerd* [21] container management system to run Kubernetes service containers and application containers. Students could use the *kubectl* [17] management utility on their virtual machine to access and manage Kind Kubernetes services running in these containers. This technology layer can be used to run programs within containers managed by Kubernetes (shown as ② in Figure 1). Laboratory assignments five and six were focused on the transition from Docker containers to the use of Kubernetes for applications running in containers. Building on the Kind Kubernetes layer, the *Kubernetes Storage* layer focused on the storage components used in Kubernetes which includes PersistentVolume and StorageClass [17]. Building on this, the *Helm* layer focused on the Helm package manager [19][18] which was used to add functionality to Kubernetes (this was one focus for the final laboratory assignment). The final *Kubernetes Node Monitoring with Prometheus and Grafana* layer was used for the final laboratory assignment.

There were several problems and challenges that needed to be addressed to use local computing resources for lab assignments for the course. These challenges included determining how to build the necessary cloud computing infrastructure from software components and determining the structure and sequence of lab assignments. Ubuntu Linux [23], Oracle VirtualBox [7] and Vagrant [8] were installed by the department IT staff on the desktop computers in the instructional lab. VirtualBox could be used through a GUI on the Ubuntu desktop to allow students to create and view virtual machines (VMs) and modify VM settings if needed for the first phase of the lab assignments.

The next section will describe the sequence of laboratory assignments and the technologies used for each assignment.

First laboratory assignment: virtual machines, Vagrant, and reproducible infrastructure

The first lab assignment asked students to create a virtual machine (VM) using Vagrant, a Vagrantfile, and an existing Vagrant Ubuntu box image. The learning objective for this assignment was focused on learning to use Vagrant, modify a Vagrantfile, and install Docker on the VM managed by Vagrant. Vagrant uses a Vagrantfile to provide configuration information and provisioning commands for the VMs created using Vagrant. An initial Vagrantfile can be created using the command *vagrant init* with the name of a vagrant box image as a command parameter. The Vagrantfile can then be edited and customized to add options, and a section of the Vagrantfile can be edited to add system commands that are run by Vagrant during VM creation. The Vagrantfile can be modified to add a virtual network interface that was bridged to a host network interface. The virtual network interface can be configured with an IP address in the Vagrantfile. Using the VirtualBox GUI, students could configure port forwarding from the Ubuntu host running on the desktop computer to a network port on their VM. Students could use port forwarding, which could then be used to *ssh* directly into their VM and access any listening ports

on their VM to test access to a running service listening on a network port. Students could add shell commands to the Vagrantfile provisioning section (either using direct commands or by invoking a script within a directory mounted to the VM) to install Docker on the VM created by Vagrant.

This lab assignment allowed students to become familiar with the use of Vagrant for creating and managing virtual machine images. This assignment emphasized reproducibility through an Infrastructure-as-Code mechanism using a Vagrantfile with the *vagrant up* and *vagrant destroy* commands. Students also gained experience with installing a Docker environment and how a container can be run on a VM. The completion of this assignment prepared students to use virtual machines (VMs) and to become aware of the transition from relying on the VirtualBox graphical user interface (GUI) to using the Vagrant command line interface (CLI) to manage VirtualBox VMs.

Second and third laboratory assignments: creating Vagrant box images and using Docker

The second lab assignment asked students to create their own Vagrant box image from an Ubuntu Linux distribution ISO image. The learning objective for this assignment was focused on learning the process involved in creating a Vagrant box image from a Linux distribution (specifically Ubuntu) and to learn how to use this box image to create a VM and to install Docker on this VM. To accomplish this, students installed Ubuntu Linux on a VirtualBox VM they created using the VirtualBox GUI on the desktop computer system. The *vagrant package* command could then be used to export the VirtualBox VM to create a Vagrant box image file that could be imported into the student's Vagrant box image cache for use as a virtual machine base image for a Vagrantfile. One motivation for this assignment was to teach students to build their own base VM images from a known software source, and to teach students about the pros and cons of relying on VM images created and published by others. This assignment also emphasized infrastructure reproducibility as a process and the need to understand the provenance of software components that are combined and integrated into a system. Completing this assignment prepared students for later assignments by learning how to build a VM image that they could customize (e.g. setting the amount of memory or the number of CPU cores) for a VM they could use for later assignments.

For the third assignment, students learned to pull and run containers using Docker and a Dockerfile, and how to mount a local directory or volume to a mount point within running NGINX [24] web server containers. The learning objective for this assignment was focused on learning to create a Docker container image and to use a shared Docker volume between containers. Since Kubernetes is a container orchestrator that manages containers, before moving to Kubernetes, students first needed to become familiar with defining, using, and managing containers. This layer provided experience for students in installing, configuring, and using Docker as a container system running on a virtual machine. Students learned about running a single container or multiple containers using a common shared persistent storage area to help students become aware of the differences between ephemeral and persistent storage. The completion of this assignment allowed students to become familiar with the process involved in creating a Docker container using a Dockerfile and how to run multiple containers with a shared Docker volume. This helped prepare students for later assignments in which they would use Kind (Kubernetes in Docker) to launch containers.

Fourth laboratory assignment: creating Docker container images and using a Docker registry

For the fourth lab assignment, students were asked to run a local Docker registry that could be used to store the container images they created. The learning objective for this assignment was focused on learning to set up and use a local Docker registry for Docker container images the students created. The assignment also asked students to preserve the image between container runs by exporting container images and importing container images from their locally running Docker registry container. The lab assignment asked students to create their own Docker container image from an Ubuntu Linux distribution using the *debootstrap* tool (the process for this is described on a website at [25]). Students learned about creating a local Docker registry for storing and distributing containers. Students built their own container images from a base Ubuntu image as an alternative to relying on prebuilt published container images that were pulled from a public container registry. This assignment provided students with experience in the topic areas of reproducibility and software provenance, and on the process needed to build their own container images. The completion of this assignment allowed students to learn to create their own container images and to store and retrieve images from a local registry as an alternative to using a pubic registry such as Dockerhub [26].

Fifth laboratory assignment: installing Kind Kubernetes nodes and running containerized applications

After gaining some experience with containers and Docker, the next step introduced students to Kubernetes as a container orchestration system. The learning objective for the fifth assignment was focused on learning to use Kind to create a functional Kubernetes installation and to run containers on a Vagrant VM the students had created in a previous assignment. For this lab assignment, students were asked to use Kind to create a one-node Kubernetes cluster and install and use the *kubectl* utility within their base VM running on a desktop computer. The kubectl command can be used to interface with and manage a running Kubernetes system [17]. To provide a bridge from using Docker for running containers to deploying containers using Kubernetes, students were asked to use the *Kompose* tool [27] to convert a Docker compose file into a set of Kubernetes YAML files for use with Kubernetes to launch within Kubernetes the containers described in the Docker compose file. Students were asked to provide the output of the *kubectl get nodes* and *kubectl get pods* commands, which can be used to show the Kubernetes nodes created using Kind and the running pods resulting from using the converted Kubernetes YAML files. This assignment built upon the previous four assignments in which students built up technology layers starting from a base system, up to a VM, on to containers running on the VM, and finally to a small running Kubernetes installation based on running Kind. For this assignment, the use of the Kompose tool to convert Docker compose files to Kubernetes YAML files provided a bridge from running the containers in Docker alone to understanding how Kubernetes could be used to orchestrate and run the containers using Kubernetes. The use of Kind allowed students to quickly and easily create a functional Kubernetes system that could immediately be used by students to explore the components of Kubernetes as well as for running containers within Kubernetes. The completion of this assignment provided students experience with creating a Kubernetes installation using Kind, and experience with starting containers within a Kubernetes

system. This provided an introduction to the basic Kubernetes elements that would be used for the remaining assignments.

Sixth laboratory assignment: creating multiple Kind Kubernetes nodes and using a LoadBalancer

Building on their basic understanding of Kubernetes from the prior assignment, the sixth laboratory assignment asked students to create multiple Kind nodes (which represent different running nodes of a Kubernetes cluster). The learning objective for this assignment was focused on learning how to create a multinode Kubernetes Kind cluster on which students could create a Kubernetes Deployment, Service, PersistentVolume, and a Kubernetes LoadBalancer to direct requests sent to a single IP address to one of multiple containers running NGINX [28]. For this Kubernetes cluster, students were asked to create a Kubernetes Deployment and Service that could use multiple Kind nodes and provide network access to Kubernetes pods running a Kubernetes Service. The application students were asked to deploy was *openresty-nginx* [28] using a Kubernetes configuration to provide more than three replicas of the NGINX containers providing the service. Students were asked to use the *echo-nginx-module* [29] to configure a 'sleep' directive to create a synthetic delay in response to HTTP requests and to use the Kubernetes LoadBalancer to spread HTTP requests across NGINX containers running on different Kubernetes nodes in Kind. Students were asked to show that their installation directed HTTP requests to different Kubernetes nodes. Students were also asked to discuss the open source license models associated with the software packages they used (primarily the *openresty-nginx* container).

The completion of this assignment provided experience for students in creating a Kubernetes Service and Deployment with a Kubernetes LoadBalancer and the use of a PersistentVolume (PV) (this prepared students for the next assignment that focused on PVs). This activity demonstrated the use of Kubernetes orchestration to scale up containers to spread a request load across multiple containers and nodes. Students also had an opportunity to become familiar with open source licensing for software packages.

Seventh laboratory assignment: Kubernetes storage

The seventh assignment asked students to create a Kubernetes Persistent Volume (PV) using an NFS server running on a separate VM they created on the desktop computer. The learning objective for this assignment was focused on learning to compare the performance and use of Docker volumes outside of Kubernetes with Kubernetes PVs. Students were asked to run a container with an application such as a Linux file system benchmark in a Docker container with a mounted volume. After this, they were asked to run the application in a container using Kubernetes with a PV to create an activity load on the mounted PV within the running container. Students were asked to discuss results from running the benchmarks within Docker containers alone (outside of Kubernetes) and running the containers within Kubernetes. Similar to a prior assignment, the assignment also asked students to discuss the open source license model associated with the software components they used.

Students completing this assignment gained experience with and awareness of: persistent storage

mechanisms within Kubernetes; using an NFS storage system with Kubernetes; and how persistent storage can be used within containers. Students also had an opportunity to explore the performance effects of using Kubernetes persistent storage mechanisms compared to running Docker containers alone. This assignment helped to prepared students for the final assignment

Final laboratory assignment: introduction to Helm and monitoring using Prometheus and Grafana

The final assignment was focused on teaching students about the use of the Helm [19] package manager for Kubernetes and the installation and use of Prometheus [22] and Grafana [30] for metric collection and the reporting of information collected from a running Kubernetes installation. The learning objective for this assignment was to learn how to install Helm packages and to install and use Prometheus and Grafana. This assignment asked students to use Helm [19] to install and start a combined installation of Prometheus and Grafana [22] using a published Helm chart named *kube-prometheus-stack* pulled from a repository [31]. Details on the process for doing this are described on a website authored by Walker [31]. Prometheus [22] gathers performance metric information from a running Kubernetes cluster. For example, the metric *alertmanager_alerts* collects the number of alerts. Grafana [30] is a tool that can be used to build monitoring dashboards (Grafana is described in Butcher [22]).

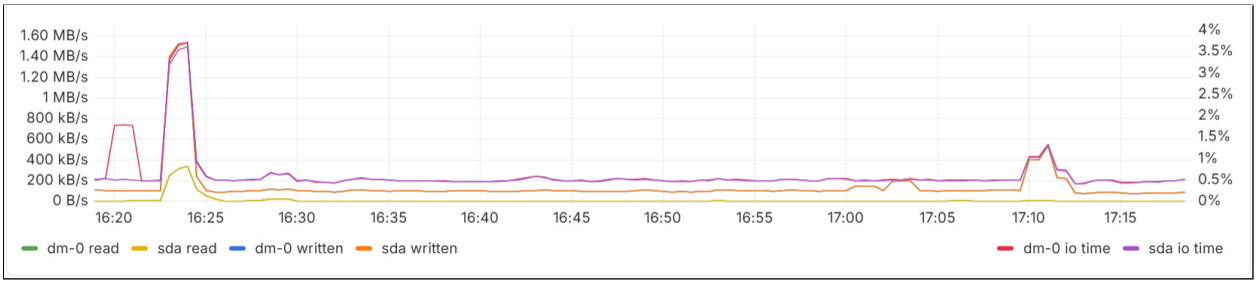
The Helm technology layer (shown in Figure 1) was focused on the use of the Helm package manager [18][19] to install Kubernetes software packages. Helm's role in Kubernetes package management is roughly analogous to (but not exactly like) the role played by software package management tools for Linux such as APT and Yum [32]. Similar to how a Linux package manager can be used to connect to a repository to download and install a collection of executable and configuration files for an application, Helm can connect to a repository to download and install a collection of files for a Kubernetes service or application to install and start application packages for Kubernetes to add functionality to Kubernetes. Helm can interact with local and remote chart repositories to use existing published Helm charts. The functions provided by Helm include the ability to pull charts from a repository, install charts, and remove (stop and uninstall) components installed by Helm.

A Helm *chart* is based on a well defined structure of directories and files that are stored in a compressed tar file. Helm is described in detail in Butcher [18]. To summarize, Helm charts include template YAML files containing embedded variables that Helm can replace with values provided in a separate *values.yaml* file that is included in the chart. Helm processes a chart to create a *manifest*, which is a collection of Kubernetes YAML files that are ready for use by Kubernetes. The manifest can be used to create the Kubernetes components, services, and pods that are needed to run an application or create a Kubernetes service. Helm charts can be *unpacked*, which is a set of directories and files representing the chart, or *packed*, which is a compressed tar file of an unpacked helm chart.

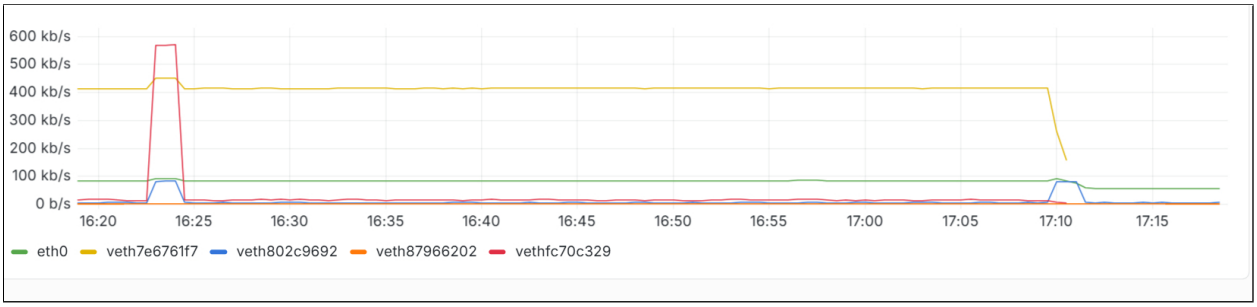
The Helm chart students used for their Kind Kubernetes cluster created a combined installation of Grafana and Prometheus (used for the top technology layer shown in Figure 1) that provided a web accessible Grafana "dashboard" accessible from a web browser that could be used to view

metrics and statistics collected by Prometheus. The dashboard can show per-node information such as CPU use, disk space use, network transmission rates, and disk I/O. To illustrate the types of information available in a Grafana dashboard, three representative panels were extracted from the screenshot of a dashboard (no benchmark was running) for clarity. The panels are shown in Figure 2. Subfigure 2a shows disk I/O information, Subfigure 2b shows network transmission information, and Subfigure 2c shows memory use information, all of which were extracted from the Grafana dashboard that is generated by running the *kube-prometheus-stack* Helm chart. As a part of the assignment, students were asked to run the file system benchmark application container they used in a prior assignment and to explore the Grafana dashboard to view the effects of running the application on some of the metrics gathered by Prometheus.

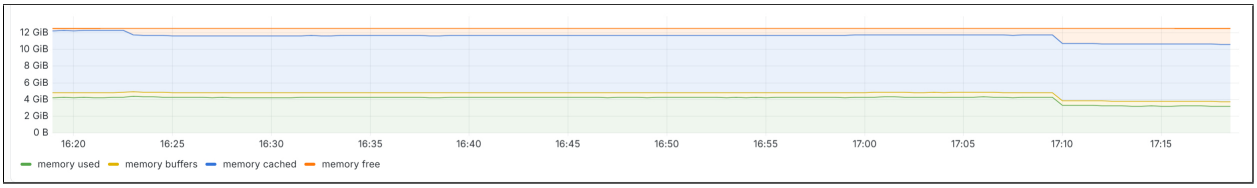
This assignment sought to build on the knowledge and skills developed by the students through the completion of all of the prior assignments. The completion of this assignment helped students to learn about the process of installing additional software components on a running Kubernetes installation and how to set up a web-based tool based on Prometheus and Grafana for monitoring a running Kubernetes installation.



(a) Disk I/O information extracted from the Grafana dashboard



(b) Network transmission information extracted from the Grafana dashboard



(c) Memory use information extracted from the Grafana dashboard

Figure 2: Examples of information displayed in a Grafana dashboard running on Kind using the *kube-prometheus-stack* [31] Helm chart.

This assignment provided students with experience installing and using a complex helm package

with two software components (Prometheus and Grafana) and awareness of Kubernetes node monitoring with Prometheus and using a Grafana based dashboard.

Reflection and some of the lessons learned

The progressive approach for the sequence of lab assignments provided students an opportunity to build up their knowledge and skills layer by layer to become familiar with containers, virtual machines, Kubernetes, and Helm. The IaC based reproducibility aspect was emphasized in three ways: through the use of Vagrant (via a Vagrantfile) for creating their VM image; creating and managing Kind based Kubernetes clusters (using a Kind configuration file); and through the use of Kubernetes YAML files for creating and starting Kubernetes services and applications within Kubernetes.

Students had an opportunity to gain experience with using the Docker container system for running containerized applications, and had an opportunity to explore the interaction between containers and Kubernetes. Students also had an opportunity to explore the differences between the Docker and Containerd container systems used by Kind for Kubernetes. By using a local Desktop computing environment for the lab assignments, students could learn the skills needed to build up their own cloud computing infrastructure based on Kubernetes without the need to rely on external cloud computing providers.

Students also had an opportunity to gain experience in building their own base virtual machine image and base containers from a Linux distribution. This activity sought to increase students' awareness of the need to understand the provenance and source of the software components embedded within VMs and container images published by others for reuse.

The desktop computers were adequate for the assignments and were able to support the software and computing load for the assignments. The computers did not include an additional NVIDIA GPU which could have been interesting for running applications in containers that could use a GPU within NVIDIA enabled Docker containers. For others seeking to develop a similar computing infrastructure for lab assignments, a recommendation is to consider the addition of a GPU within a desktop computer that could be used for assignments for GPU enabled applications.

Conclusions

This paper described a progressive sequence of lab assignments for a cloud computing class conducted during the Spring 2025 semester. These assignments allowed students to build up a Kubernetes based cloud computing infrastructure that could use desktop scale computers in a teaching laboratory for their laboratory assignments without the need to use resources from an external commercial cloud provider. Students completing the assignments had an opportunity to gain skills and experience with Infrastructure-as-Code and reproducibility using Vagrant and Kind, managing and using containers using Docker, awareness of another container system (Containerd) that is used within Kind for running Kubernetes, Kubernetes components such as persistent storage and services, and the Helm Kubernetes package manager. The desktop scale teaching laboratory computer and local resources used by each student were adequate for the

laboratory assignments. A recommendation for others seeking to explore a similar approach for a cloud computing infrastructure class would be to consider adding a GPU to the desktop lab computer for use by GPU enabled applications running in containers orchestrated by Kubernetes.

References

[1] "Kind," 2025, Accessed November 21, 2025. [Online]. Available: https://kind.sigs.k8s.io/

[2] B. Burns, J. Beda, K. Hightower, and L. Evenson, *Kubernetes: up and running: dive into the future of infrastructure*. O'Reilly Media, Inc., 2022.

[3] R. Podeschi and J. DeBo, "Integrating aws cloud practitioner certification into a systems administration course." *Information Systems Education Journal*, vol. 20, no. 5, pp. 17–26, 2022.

[4] J. Strain, J. Marquardson, and D. L. Gomillion, "Conquer the cloud: Effectively using the amazon web services academy learner lab for information systems education." *Information Systems Education Journal*, vol. 23, no. 6, 2025.

[5] "2023's top resources for higher education that cloud providers offer for free," 2023, Accessed on November 30, 2025. [Online]. Available: https://kion.io/2023s-top-resources-for-higher-education-that-cloud-providers-offer-for-free/

[6] K. Morris, *Infrastructure as code, Third Edition*. O'Reilly Media, 2025.

[7] "Virtualbox," 2025, Accessed November 24, 2025. [Online]. Available: https://www.virtualbox.org/

[8] "Vagrant," 2025, Accessed November 24, 2025. [Online]. Available: https://developer.hashicorp.com/vagrant

[9] D. Cutting, E. L. De Pina, E. Barlaskar, A. McDowell, and N. Anderson, "A toolchain for the in-house education of cloud computing in a vendor agnostic and efficient manner," in *2025 IEEE Engineering Education World Conference (EDUNINE)*, 2025, pp. 1–6.

[10] E. H. Salib, "Empowering undergraduate students with cloud computing skills: A proposal for openstack-centric education," in *2024 IEEE Frontiers in Education Conference (FIE)*, 2024, pp. 1–9.

[11] F. Xue, L. Yang, B. Fan, H. Jiang, and Y. Huo, "Kube-cc: Building an experiment education platform for big data courses based on kubernetes," in *2023 International Conference on Engineering Education and Information Technology (EEIT)*, 2023, pp. 14–18.

[12] R. L. Neupane, P. Calyam, S. Wang, K. Neupane, A. Pandey, X. Cheng, D. Gafurov, H. S. Yeddulapalli, N. Glaser, K. P. Singh, Y. Gu, S. Li, and S. Srinivas, "Online self-service learning platform for application-inspired cloud development and operations (devops) curriculum," *IEEE Transactions on Learning Technologies*, vol. 17, pp. 1906–1920, 2024.

[13] D. Sedov and A. Lazarev, "Enhancing it education through docker integration," in *2024 4th International Conference on Technology Enhanced Learning in Higher Education (TELE)*, 2024, pp. 252–255.

[14] O. S. Holovnia and V. Oleksiuk, "Selecting cloud computing software for a virtual online laboratory supporting the operating systems course," in *CTE 2021: 9th Workshop on Cloud Technologies in Education, December 17, 2021, Kryvyi Rih, Ukraine*, no. 3085. CEUR Workshop Proceedings, 2022, pp. 216–227.

[15] N. Chouliaras, I. Kantzavelou, L. Maglaras, G. Pantziou, and M. A. Ferrag, "A novel autonomous container-based platform for cybersecurity training and research," *PeerJ Computer Science*, vol. 9, p. e1574, 2023.

[16] Rancher, "Rancher," https://www.rancher.com/, accessed: 2026-02-16.

[17] J. Vyas and C. Love, *Core Kubernetes*. Manning Publications, 2022.

[18] M. Butcher, M. Farina, and J. Dolitsky, *Learning Helm*. O'Reilly Media, Inc., 2021.

[19] “Helm,” 2025, Accessed November 18, 2025. [Online]. Available: https://helm.sh/

[20] “Docker,” 2025, Accessed November 24, 2025. [Online]. Available: https://www.docker.com/

[21] “containerd an industry-standard container runtime with an emphasis on simplicity, robustness and portability,” 2025, Accessed on November 24, 2025. [Online]. Available: https://containerd.io/

[22] J. Pivotto and B. Brazil, *Prometheus: Up & Running*. O’Reilly Media, Inc., 2023.

[23] “Cannonical ubuntu,” 2025, Accessed December 8, 2025. [Online]. Available: https://ubuntu.com/

[24] “Nginx,” 2025, Accessed November 25, 2025. [Online]. Available: https://nginx.org/

[25] “Docker base images,” 2025, Accessed November 25, 2025. [Online]. Available: https://docs.docker.com/build/building/base-images

[26] Dockerhub, 2026, accessed February 12, 2026. [Online]. Available: https://hub.docker.com/

[27] “Kompose,” 2025, Accessed November 24, 2025. [Online]. Available: https://kompose.io/

[28] “Openresty,” 2025, Accessed November 25, 2025. [Online]. Available: https://hub.docker.com/r/openresty/openresty

[29] “echo-nginx-module,” 2021, Accessed November 25, 2025. [Online]. Available: https://github.com/openresty/echo-nginx-module

[30] “Grafana,” 2025, Accessed November 24, 2025. [Online]. Available: https://grafana.com/

[31] J. Walker, “Prometheus monitoring for kubernetes cluster [tutorial],” 2025, Accessed November 25, 2025. [Online]. Available: https://spacelift.io/blog/prometheus-kubernetes

[32] “Helm charts: making it simple to package and deploy common applications on kubernetes,” 2016, Accessed November 24, 2025. [Online]. Available: https://kubernetes.io/blog/2016/10/helm-charts-making-it-simple-to-package-and-deploy-apps-on-kubernetes/