Leveraging Chameleon Cloud for Hands-On Learning in Systems and Data Courses: An Experience Report

Fraida Fund
ffund@nyu.edu
New York University
Brooklyn, New York, USA

Erez Zadok
ezk@cs.stonybrook.edu
Stony Brook University
Stony Brook, New York, USA

Tyler Estro
testro@cs.stonybrook.edu
Stony Brook University
Stony Brook, New York, USA

David Koop
dakoop@niu.edu
Northern Illinois University
DeKalb, Illinois, USA

Kate Keahey
keahey@mcs.anl.gov
Argonne National Laboratory
Lemont, Illinois, USA

Marc Richardson
mtrichardson@uchicago.edu
University of Chicago
Chicago, Illinois, USA

Abstract

Large-scale hands-on computing education often requires infrastructure beyond what individual courses can provide. Research infrastructure can fill this gap. We present an experience report on using the [Anonymous Testbed] in four course offerings, organized into three cases: storage systems, data analysis and visualization, and machine learning systems engineering and operations. Across these cases, [Anonymous] supported realistic, reproducible, and flexible learning experiences, while also introducing operational challenges around workflow design, support, and scale. We conclude with practical lessons for instructors designing systems- and data-intensive courses around shared research infrastructure.

CCS Concepts

• **Social and professional topics** → **Computing education**; • **Computing methodologies** → **Machine learning**; • **Networks** → **Cloud computing**.

Keywords

computing education, cloud computing, research infrastructure, computer systems education

ACM Reference Format:
Fraida Fund, Erez Zadok, Tyler Estro, David Koop, Kate Keahey, and Marc Richardson. 2018. Leveraging Chameleon Cloud for Hands-On Learning in Systems and Data Courses: An Experience Report. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

As computing and data-intensive methods become central to science, engineering, and society, there is a growing need for educational experiences that prepare students to work with large-scale, production-like infrastructure. Coursework on topics such as high-performance computing, distributed systems, cloud computing, at-scale data processing and visualization, data storage systems, and machine learning systems engineering form an increasingly important part of the computing curriculum at both the undergraduate and graduate level. In the age of AI in particular, it is increasingly important to teach and assess students on working with realistic, large-scale systems rather than only small toy problems. However, teaching these courses effectively requires access to infrastructure that can support realistic, complex scenarios, often beyond what is reasonable for an individual instructor to provide and maintain.

To address this problem, educators may use a commercial cloud service, have students use their own devices, use a platform specific for education in a particular topic, or build and maintain their own institutional infrastructure. Each of these options has significant limitations. The cost - and particularly the uncertainty associated with the cost - for use of commercial cloud services can be a deterrent for many educators. Using students' personal devices raises concerns about equity, requires substantial effort on the part of the educator to troubleshoot a variety of personal compute environments, and does not necessarily support all of the required scenarios. Platforms designed specifically for education (e.g. DeterLab for cybersecurity [13]) offer prepared materials that are tightly aligned with infrastructure, but it is often not possible or very difficult for educators to adapt or to go significantly beyond the materials provided. Finally, self-managed infrastructure is expensive, effort-intensive to build, and requires ongoing investment to manage hardware and software rot.

Fortunately, research infrastructure can often address this need. Research infrastructure has long been a powerful tool for the computer science classroom [4, 11, 15, 17, 20, 22–24]. The [Anonymous] [3] platform used by the educators in this report, in particular, supports a range of complex research experiments in cloud computing, high performance computing, machine learning, edge computing, and storage systems. It therefore provides a highly capable environment for *teaching* these subjects. Its capabilities are well-used and battle-tested in an education setting, having been used to teach high performance computing, cloud computing, machine learning, and autonomous driving, in classroom settings as well as less traditional education and training settings.

Conference acronym 'XX, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

In this paper, we present an experience report on four course offerings that integrated [Anonymous] for hands-on learning, organized into three cases with different subject areas and instructional goals. While all four offerings leverage the same research infrastructure, they do so in unique ways shaped by their disciplinary focus, pedagogical goals, and institutional context. One case (**C1**), a special topics course at a large public state flagship university, introduces students to the design and implementation of *storage systems*, emphasizing scalability, reliability, and performance. A second case (**C2**) spans courses on *data analysis and visualization* at a public R2 institution, giving students hands-on practice with transforming complex datasets into interpretable insights through interactive tools and platforms. A third case (**C3**), a course on *machine learning systems engineering and operations* offered to graduate students at a large private R1 university, engages students with the infrastructure required to train, deploy, and monitor large-scale ML systems. Together, these cases illustrate the flexibility of shared research infrastructure to support diverse educational contexts. The goal of this report is to reflect on the experience of using [Anonymous], including the tradeoffs (relative to other infrastructure options) involved in cost, control, usability, and effort. By doing so, we hope to provide insights that can guide instructors in adopting research infrastructure effectively in their own classrooms.

Our contributions are threefold. First, we describe the context, materials, and classroom experiences of each case, highlighting the ways in which [Anonymous] was deployed in teaching. Second, we identify the opportunities and challenges of using research infrastructure at scale. Finally, we note cross-cutting lessons about how shared infrastructure can enable authentic, hands-on learning across multiple domains and institutions.

The courselet materials described in this report are publicly available on [Anonymous] [links removed for double blind review].

2 Related work

Prior work has shown that research testbeds can be effective for hands-on computing education across several topic areas. Instructors have used GENI and FABRIC to teach computer networks [8, 10, 11, 16, 20, 22]. For cybersecurity-focused courses, instructors have used CloudLab and GENI-based environments for network security labs [14–17, 21], in addition to the specialized DETER [13] infrastructure. In cloud computing, instructors have used GENI and Emulab for cloud lab exercises and course activities [1, 4, 23].

However, the dominant emphasis in this literature is on networking, cloud computing, and network security education. In contrast, our report focuses on how research infrastructure can support computer systems courses that are not represented in this line of prior work: data systems, machine learning systems, and storage systems. Our contribution is therefore not just a platform-use case, but a cross-course experience report showing how shared research infrastructure can support systems- and data-intensive learning objectives beyond the traditional networking/cloud/security scope of research testbeds.

3 Background: [Anonymous] testbed

[Note: PC asked us to anonymize the testbed name; so we had to remove citations that reference its name.]

The courses described in this report use the Chameleon Cloud research testbed [12]. Chameleon Cloud is an NSF-funded platform for experimental computer science research, education, and emergent applications. It is based on the open-source OpenStack cloud services and is distributed across three hardware sites at supercomputing centers in Chicago, IL, Austin, TX, and Cheyenne, WY. It provides bare metal and virtualized compute, networking, and storage resources, supporting a wide variety of activity across the edge to cloud research continuum. Since 2014, Chameleon has served 14,500+ users working on 1,300+ projects resulting in 1,000+ publications. While a majority of these are research projects, the system has served many education projects [2, 5–7, 9, 18, 19], including undergraduate and graduate courses with interactive assignments and class projects; conference tutorials; and educational hackathons and competitions. It is free to use for researchers, educators, and students.

The instructors represented in this report found [Anonymous] especially powerful for education because it combines diverse hardware with flexible workflows. Instructors can choose among virtual machines, bare-metal servers, nodes with GPUs, FPGAs, or NVMe, and edge devices such as Raspberry Pi and Jetson boards.As an OpenStack-based cloud, [Anonymous] also provides cloud networking and storage primitives such as private virtual networks, floating IP addresses, block storage volumes, object storage buckets, and NFS file-sharing services.

[Anonymous] can be used through standard tools such as Terraform and the OpenStack CLI, or through [Anonymous]-specific interfaces. Its hosted Jupyter service lets users launch a browser-based Jupyter container and use its `python` API to provision resources, execute commands on them, and carry out experiments.

[Anonymous] also includes [Anonymous], a sharing portal for packaging notebooks, scripts, configuration files, and instructions as reproducible artifacts that students can launch directly into a [Anonymous]-hosted Jupyter environment.

For educators, the main design choices are how to enroll students, reserve and configure resources, distribute materials, and decide how students will interact with the resource. These choices (summarized in Table 1) depend on both pedagogy and logistics. If resource management is itself a learning objective, students may reserve, launch, and configure resources using the GUI, CLI, APIs, or notebooks; if it is background setup, course staff may reserve resources in advance or hide complexity behind prepared scripts. Similarly, SSH is appropriate when students should learn standard system tools, while browser-based notebooks can reduce friction when the goal is to run experiments and interpret results. Scarce or shared resources require more instructor coordination, while on-demand virtual machines and open-ended projects can support more student-managed workflows.

In the next section, we describe the use of [Anonymous] in four computer systems and data systems course offerings, grouped into three cases with different choices and tradeoffs.

Compute Resources	Project Enrollment	Provisioning Resources	Distributing Materials	Interacting with Resources
General purpose compute	Individual add/remove	GUI (OpenStack Horizon)	[Anonymous] sharing portal	SSH (with keys)
FPGAs, GPUs, NVMe	Bulk add/remove	OpenStack client	Git workflow	JupyterHub on resource
Edge devices	"Join" link	[Anonymous] `python` API library	Other (e.g., LMS)	[Anonymous] JupyterHub

Table 1: The "menu" of logistical choices for prospective educators on [Anonymous].

4 Experiences: Using [Anonymous] for Education

As introduced earlier, this study examines four course offerings - one on storage systems (**C1**), two on data analysis and visualization (**C2**), and one on machine learning systems engineering and operations (**C3**) - that integrated [Anonymous] as a platform for experiential learning. Each case embodies a distinct perspective on how research infrastructure can enhance computing education: **C1** emphasizes safe, repeatable, low-level experimentation with storage components and failure scenarios, along with student contribution to reusable instructional artifacts; **C2** focuses on reproducible, instructor-managed environments for data analysis and visualization workflows that can evolve with rapidly changing tools while supporting hands-on practice; and **C3** centers on the design of machine learning systems across their full lifecycle, including the compute, storage, network, and systems layers that underpin modern ML infrastructure. In the following sections, we describe how [Anonymous] was incorporated into each case, the instructional and logistical choices made by the instructors, and how these choices reflect the pedagogical priorities and institutional contexts of the four offerings.

4.1 C1: Storage Systems

4.1.1 Context. **C1** was a graduate-level special topics offering on storage systems at Anonymous Institution, with 20 students enrolled. Its goal was to give students hands-on experience with the design and implementation of storage systems, with particular emphasis on scalability, reliability, and performance. In a course of this kind, many of the most valuable exercises are difficult to teach through lectures alone. Students need to manipulate disks, partitions, logical volumes, file systems, and network storage directly, (all with superuser privileges)—and they benefit from seeing how those components behave under realistic administrative actions, performance experiments, and failure scenarios.

Storage education also poses a practical challenge: many relevant experiments are slow, stateful, and sometimes destructive. It is useful to let students build small proof-of-concept configurations using tiny virtual disks, because the same concepts can then be explored more quickly and repeatedly than on larger physical systems. At the same time, some scenarios, such as degraded arrays or corruption-style failures that could lead to data loss, are impossible to run on live systems. These characteristics make storage courses especially well suited to research infrastructure that can be reconfigured easily and reset to a blank slate when needed.

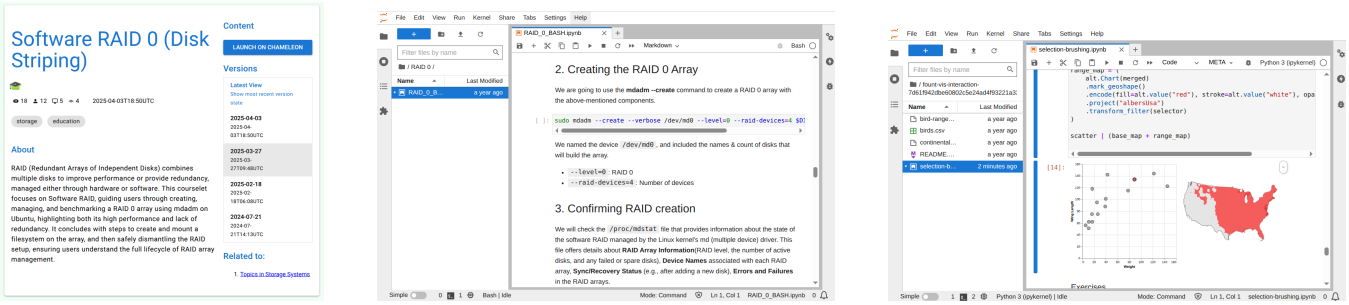
Before adopting [Anonymous], past iterations of the course relied first on physical infrastructure managed by the course staff, and later on local virtualization platforms running on student's own devices. Physical setups were constrained by the number of disks available and made it difficult to give every student an independent environment for experimentation. Later iterations used VMware for part of this workflow, then moved to Proxmox after Broadcom cut off its education use. Although Proxmox proved to be a strong platform, it still took more effort than desired to ramp students up consistently and reproduce the same environment across the course. [Anonymous] changed this instructional model by providing isolated virtual environments that students could provision themselves, configure with the required number of storage volumes, and reset whenever an experiment needed to be repeated from scratch.

A second instructional goal was to make the course materials themselves more reusable. In systems courses, instructors often invest substantial effort building demonstrations and labs that remain tied to a single semester or to local infrastructure. We wanted instead to develop artifacts that could be refined and reused beyond the course itself, both in future offerings and as onboarding material for student researchers.

4.1.2 Materials. The course used *courselets* - small modules for interactive exploration of a single topic - as the main unit of hands-on instruction. We began with five instructor-authored courselets that established a common baseline for all students: *Initializing a VM at KVM@TACC With Multiple Storage Volumes*, *Disk Partitioning*, *Benchmarking Disk I/O Performance*, *Software RAID 0 (Disk Striping)*, and *Software RAID 1 (Disk Mirroring)*. These courselets were published on [Anonymous]. The first included an executable notebook for provisioning and configuring the required [Anonymous] resources with the [Anonymous] `python` API, the OpenStack Python API, and the OpenStack GUI; the remaining notebooks explained each activity and guided students through interacting with the system over SSH.

These materials introduced students to the environment, taught them how to request and configure virtual machines and storage volumes, and familiarized them with the format and expectations of the courselets. They also served as templates for later in the course, when students are tasked with creating their own materials.

After this onboarding phase, students were asked to design their own storage courselets using the same notebook-based format and [Anonymous]-based workflow. Each team or individual was assigned a storage topic and was expected to build a courselet that was technically correct, clearly documented, and accessible to a novice audience. The instructional staff reviewed draft versions, provided feedback during development, and curated and cleaned up the strongest submissions for publication. This shifted students from being only users of instructional content, to active contributors to a reusable teaching corpus.



(a) Launching a courselet. (b) Sample storage courselet. (c) Sample data visualization courselet.

Figure 1: Example courselet materials from the storage systems and data visualization courses.

The resulting set of published courselets consisted of the five instructor-authored artifacts above together with 12 new, student-developed courselets on *Understanding Symbolic Links*, *Understanding Hard Links*, *Logical Volume Manager*, *Data Compression*, *Data Deduplication and Compression*, *Data Integrity with dm-integrity*, *Disk Encryption with dm-crypt*, *Btrfs File System*, *XFS File System*, *NFSv3*, *NFSv4*, and *Storage Caching*. In total, 17 courselets were published on [Anonymous].

4.1.3 Experience. While we did not undertake a formal evaluation of these activities, to understand students' experience, we collected survey feedback after the initial courselets. Responses were broadly positive. Students generally reported that the courselets helped them learn new concepts and improved their understanding, and the feedback was especially favorable for the disk partitioning material. At the same time, the responses also highlighted a real cost of this approach: early setup and infrastructure onboarding could feel somewhat cumbersome, particularly before students became comfortable with the environment.

That tradeoff was acceptable, and in some cases desirable, for this course. A storage systems class should not completely hide the mechanics of provisioning resources, attaching disks, configuring devices, and recovering from mistakes, because these actions are part of the subject matter itself. What [Anonymous] provided was not the elimination of friction, but a form of friction that was pedagogically useful: students could make mistakes, break configurations, and rerun experiments without jeopardizing a shared production system or exhausting scarce physical hardware.

More broadly, the course showed that research infrastructure can support storage education in two complementary ways. First, it enables safe, repeatable, and reconfigurable experiments that would otherwise be too slow, too destructive, or too cumbersome to offer at scale. Students could use tiny virtual disks as proof-of-concept stand-ins for larger systems, and instructors could demonstrate scenarios such as RAID-style degraded or failure conditions without placing real data at risk. Second, the same infrastructure can serve as a framework for producing durable educational artifacts rather than merely hosting one semester's assignments. In this course, [Anonymous] was not only the environment in which students learned storage concepts, but also the medium through which they helped create a reusable set of courselets for future students and new lab members.

4.2 C2: Data Analysis & Visualization Courses

4.2.1 Context. **C2** covers two courses offered at a public R2 institution: *Advanced Data Management* and *Data Visualization*.

Courses in data analysis and visualization often combine conceptual and applied learning, and both courses balance these components by integrating lectures and discussion with significant programming assignments and projects. While all content changes over time, changes in technology can often require large changes. For example, `pandas` had been the dominant platform for data analysis in Python for a decade, but newer frameworks like `polars` have advantages like multithreading and a more consistent API. Thus, tightly integrating specific implementations with course content can be harder to maintain. In addition, having examples to follow can be important both for learning concepts more concretely but also for understanding the implementation details. These can run in different timelines than other material updates and can be more easily shared among instructors.

4.2.2 Materials. The Data Visualization course used four courselets, including chart development in two different libraries, colormapping, and interactive processing. The Advanced Data Management course used five courselets, with materials on leveraging data frameworks like `pandas`, `polars`, and DuckDB, as well as on topics like data cleaning, data transformation, and data integration.

In the past, these courses have evolved across different tools and frameworks, beginning with the core D3 library in textual HTML and Javascript source files, then moving to Observable notebooks to mitigate the need for distributing multiple files. Those were hosted on a proprietary platform. Moving to [Anonymous] allowed materials to be shared and redistributed more easily.

4.2.3 Experience. To better understand how students experience courselets, course staff distributed an anonymous online survey. Students were asked about each of four courselets: Data Visualization using `matplotlib`, Data Visualization using `obsplot`, Color in Data Visualization, and Selection & Brushing in Data Visualization. For each courselet, students rated whether it helped them learn new concepts, improved their understanding, increased their confidence, made the topic more engaging, and supported future application. They also compared the helpfulness of courselets with other learning resources, including assignments, lectures, online resources, in-class examples, AI tools, and the textbook. Because

this survey was designed for ongoing improvement of course materials and not as a formal study, we report general trends rather than exact quantitative results.

In general, students reported that they perceived courselets as helping them learn and understand the materials. They gave lower ratings for increased confidence, perhaps indicating that students realized there was still room for improvement. Standard deviations were relatively low, reflecting tight clustering of responses and consistent perceptions across students. Overall, these results show that each topic was perceived as engaging and educational, with only minor differences in the question answers.

When comparing different types of learning resources, students found most materials helpful, but lectures, assignments, in-class examples, and courselets consistently received higher ratings. Online materials and AI tools were viewed positively but a bit lower than the traditional resources. The textbook received the lowest ratings. While this may parallel a trend where students consult textbooks less frequently now than in the past, the textbook for this course focused on the theory of data visualization and did not focus on instructing students on how to code or create the visualizations using visualization tools or libraries.

4.3 C3: Machine Learning Systems Engineering and Operations

4.3.1 Context. Finally, **C3** *Machine Learning Systems Engineering and Operations* was developed in response to growing demand for engineers who can design, deploy, and operate ML systems beyond the model layer. The course treated machine learning as a full-stack systems problem spanning compute, storage, networking, and operations, with realistic infrastructure serving as both the medium and the subject of instruction. It was first offered in Spring 2025 as a graduate special topics course at a large private R1 university, enrolling 191 students, primarily master's students in Electrical and Computer Engineering with some participation from Computer Science, Data Science, and Mathematical Sciences.

4.3.2 Materials. The major instructional component of the course was a sequence of 14 hands-on courselets, structured as weekly lab assignments that built progressively toward end-to-end ML systems. Lectures introduced the system-level concerns for each unit, but students practiced the central learning objectives via courselets: designing, building, deploying, evaluating, and operating ML systems across their lifecycle. Together, the courselets accounted for sixty percent of the grade. A final project made up the remaining forty percent, and required groups to build a medium- to large-scale ML system, with students taking ownership of subsystems such as training, serving, data pipelines, or DevOps.

The courselets made infrastructure engagement a core learning activity rather than background setup. Students worked directly with compute, network, storage, and systems layers using Python APIs, command-line clients, web interfaces, Terraform, SSH, Ansible, Kubernetes, CPU and GPU instances, and edge devices. Many courselets were unified by a semester-long case study, *GourmetGram*, a fictional photo-sharing service that required students to deploy, scale, and operate ML-based services. This recurring application context helped connect individual infrastructure tasks to the larger ML systems lifecycle.

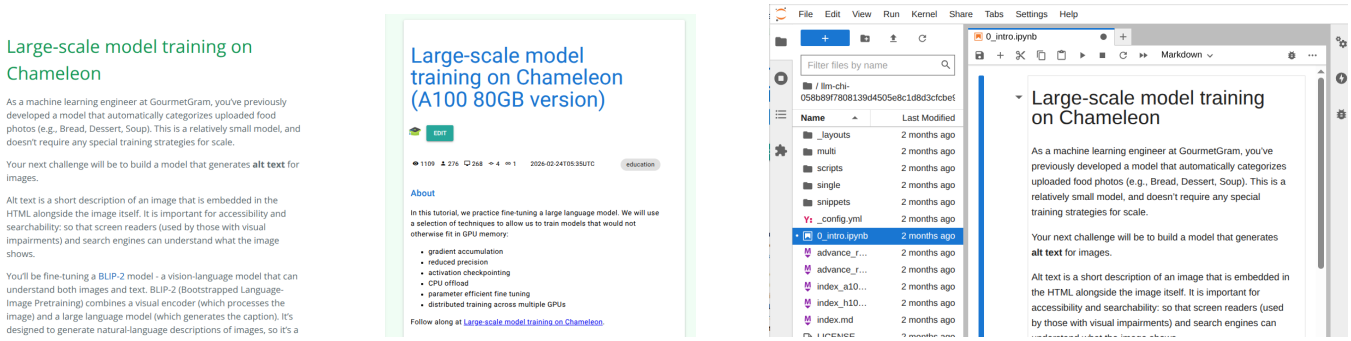
The sequence began with [Anonymous] onboarding, VM provisioning, and SSH access, then introduced cloud computing concepts through VMs, networks, floating IPs, containers, and Kubernetes. Subsequent courselets moved from manual setup to DevOps automation with Terraform, Ansible, Argo CD, and Argo Workflows; data systems using block storage, object storage, and data pipelines; model training at scale on one or more GPUs; experiment tracking with MLflow; distributed training with Ray; serving with ONNX and NVIDIA Triton; and monitoring and evaluation using behavioral tests, fairness checks, slice-based metrics, drift detection, and operational metrics. Later activities connected the [Anonymous]-based workflow to project work and, optionally, to a commercial cloud environment, emphasizing transferable operational skills rather than platform-specific procedures.

The courselets also served as preparation for the final project. By the time students began project work, they had practiced the main operational patterns needed for larger systems: provisioning infrastructure, automating deployment, managing data and storage, training and serving models, and monitoring behavior after deployment. The project then required students to recombine these patterns in a less prescriptive setting, making the courselets both instructional materials and a bridge to independent systems work.

The materials were authored in Markdown and built with a Makefile into two complementary formats: a lightweight GitHub Pages site and an interactive notebook version published through [Anonymous]. This allowed students to browse the materials quickly for reference, or launch the same courselets in an executable, testbed-integrated environment. Since materials are hosted in Github, instructors who want to adapt them can easily fork them and make changes compatible with their own local circumstances.

4.3.3 Experience. **C3** had a large enrollment, compared to **C1** and **C2**, which were much smaller in scale. The scope of [Anonymous]'s involvement in this course was also much larger: in the Spring 2025 offering, students used more than 180,000 compute-instance hours, or almost 1,000 compute-instance hours per enrolled student, and over 10TB of storage. (Because some activities involved clusters with multiple simultaneous compute instances, compute hours does not correspond directly to student time.) This level of activity would have been difficult to support on other platforms: using a cloud cost calculator, we estimate that the cost of the course on GCP or AWS would have been approximately $50,000.

This exposed some scaling issues in using shared research infrastructure for teaching. Ensuring access to scarce resources, such as multi-GPU instances, in burst mode (for example, all students running a multi-GPU training experiment in the same week) required advance planning and coordination between the course staff and the [Anonymous] team through the help desk ticket system. Because some materials were tied to or calibrated for specific hardware, course staff also had to develop and maintain multiple parallel versions of some activities, such as NVIDIA GPU and AMD GPU versions, so that enough students could access compatible resources on [Anonymous]. Bugs and operational limits also surfaced at this scale, both within and beyond [Anonymous] itself. For example, shared networks encountered IP address exhaustion in configurations that had not previously accommodated a class of this size, and Docker pulls were temporarily blocked when many



(a) Lightweight website version. **(b) [Anonymous] artifact page.** **(c) Interactive notebook version.**

Figure 2: Sample machine learning systems courselet: a GitHub Pages website for lightweight browsing, a [Anonymous] artifact page for launching the activity, and the corresponding interactive notebook for execution in [Anonymous].

students pulling images through the same shared IP address exhausted Docker Hub rate limits.

Supporting this scale required substantial human infrastructure, including weekly office hours with the instructor and course assistants and an online Q&A forum that accumulated over 700 discussion threads and more than 3,000 posts during the semester.

Feedback indicated strong educational value as perceived by students: in end-of-semester evaluations, almost all students reported that the infrastructure-based approach added "very significant" or "somewhat significant" value to their learning, and rated the lab materials as supportive. Anecdotally, as with **C1**, the infrastructure introduced real friction that sometimes frustrated students, as evidenced by their posts on the Q&A forum; however, this friction was in most cases pedagogically useful because students were learning to diagnose and recover from the kinds of configuration, deployment, and resource-management problems that arise in real ML systems work. This became especially evident in the later parts of the course, when students had to design, build, and debug their own large-scale ML systems, and were able to apply the skills developed through the guided courselets.

5 Lessons learned and recommendations

Across the three cases, shared research infrastructure enabled realism that would have been difficult to provide with conventional course infrastructure. In **C1**, [Anonymous] supported destructive and repeatable storage experiments in isolated environments; in **C2**, it provided reproducible environments for data analysis and visualization workflows that evolve; and in **C3**, it supported large-scale, cloud-native, GPU-enabled ML systems activities.

A second lesson is that instructors should intentionally decide how much of the infrastructure workflow students should see. When setup, provisioning, debugging, or direct use of system tools is part of the learning goal, as in **C1** and **C3**, exposing students to SSH, command-line tools, cloud APIs, and resource configuration can be pedagogically valuable even when it introduces friction. When the goal is to focus attention on analysis, visualization, or interpretation, as in parts of **C2**, lower-friction workflows such as [Anonymous]-launched notebooks may be more appropriate.

Scale requires planning, coordination, and human support. The largest offering, **C3**, showed that burst demand for scarce resources, shared-network limits, third-party service rate limits, and student support load can become significant when many students use research infrastructure simultaneously. Instructors planning large courses should coordinate early with infrastructure operators, reserve scarce resources, stage deadlines when possible, prepare fallbacks for external-service limits, and budget staff time for office hours and asynchronous support.

Finally, course materials should be reusable artifacts. [Anonymous] made it possible to package notebooks, scripts, configuration files, and instructions into materials that students could launch reproducibly and instructors could revise across offerings. In **C1**, students also contributed new courselets, turning course work into a reusable teaching corpus. Instructors should treat infrastructure-based assignments not only as one-time labs, but as artifacts that can be shared, adapted, and improved over time.

6 Conclusions

This experience report described four course offerings, organized into three cases, that used [Anonymous] for hands-on learning in storage systems, data analysis and visualization, and machine learning systems engineering and operations. Across these cases, shared research infrastructure enabled students to work with realistic systems, from low-level storage configurations to reproducible data workflows and large-scale ML deployments, and student feedback was generally positive.

The flexibility of large-scale research infrastructure comes with operational complexity. Our experience suggests that this complexity can be worthwhile when instructors align infrastructure workflow with learning goals, provide appropriate support, and package materials for reuse. With these choices, shared research infrastructure can make authentic systems- and data-intensive learning feasible across diverse courses and institutions.

Acknowledgments

This material is based upon work supported by the U.S. National Science Foundation under Grant Nos. 2230077, 2230078, 2230079, and 2431425.

References

[1] Jay Aikat, Michael K. Reiter, and Kevin Jeffay. 2016. Education Modules for Networking, Cloud Computing, and Security in Systems Courses. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) *(SIGCSE '16)*. Association for Computing Machinery, New York, NY, USA, 400. doi:10.1145/2839509.2850514

[2] Richard Anderson. 2022. *Driving Autonomous Cars From Edge to Cloud with CHI@Edge.* https://chameleoncloud.org/blog/2022/12/19/driving-autonomous-cars-from-edge-to-cloud-with-chiedge/

[3] Anonymous author. [n. d.]. PCs asked us to anonymize testbed name. In *Proceedings of the Anonymous Conference.*

[4] Prasad Calyam, Sripriya Seetharam, and Ronny Bazan Antequera. 2014. GENI Laboratory Exercises Development for a Cloud Computing Course. In *2014 Third GENI Research and Educational Experiment Workshop*. 19–24. doi:10.1109/GREE.2014.15

[5] Massimo Canonico. 2023. *Presentation for mini-symposium on education using Chameleon.* Video available in program listing at: https://chameleoncloud.org/chameleon-cloud-users-meeting/user-meeting-2023/.

[6] Priyanka Bose Chandra Shekhar Pandey. 2023. *Reproduce, Rerun, Repeat: The Fun Way to Learn Machine Learning!* https://www.chameleoncloud.org/blog/2023/03/28/reproduce-rerun-repeat-the-fun-way-to-learn-machine-learning/

[7] Jose Manuel Monsalve Diaz. 2023. *Using Chameleon for HPC Education: an OpenMP Tutorial.* https://www.chameleoncloud.org/blog/2023/05/30/jose-monsalve-education-user-blog-post-interview/

[8] Fraida Fund and Shivendra S. Panwar. 2020. Teaching Computer Networks with GENI. In *Proceedings of the 2020 ACM SIGCOMM Education Workshop.*

[9] Aniruddha Gokhale. 2023. *Educating with Chameleon at Vanderbilt.* https://www.chameleoncloud.org/blog/2023/07/17/educating-with-chameleon-at-vanderbilt/

[10] James Griffioen, Zongming Fei, Hussamuddin Nasir, Xiongqi Wu, Jeremy Reed, and Charles Carpenter. 2012. Teaching with the emerging GENI network. In *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing, 1.

[11] James Griffioen, Zongming Fei, Hussamuddin Nasir, Xiongqi Wu, Jeremy Reed, and Charles Carpenter. 2013. GENI-Enabled Programming Experiments for Networking Classes. In *2013 Second GENI Research and Educational Experiment Workshop*. 111–118. doi:10.1109/GREE.2013.30

[12] Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S. Gunawi, Cody Hammock, Joe Mambretti, Alexander Barnes, François Halbach, Alex Rocha, and Joe Stubbs. 2020. Lessons Learned from the Chameleon Testbed. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20).*

[13] Jelena Mirkovic and Terry Benzel. 2012. Teaching cybersecurity with DeterLab. *IEEE Security & Privacy* 10, 1 (2012), 73–76.

[14] Xenia Mountrouidou and Vic Thomas. 2019. CyberPaths: Cyber Security Labs for Liberal Arts Institutions Using the NSF Global Environment for Network Innovations (GENI). In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 1241. doi:10.1145/3287324.3287541

[15] Younghee Park, Hongxin Hu, Xiaohong Yuan, and Hongda Li. 2018. Enhancing Security Education Through Designing SDN Security Labs in CloudLab. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 185–190. doi:10.1145/3159450.3159514

[16] Mohamed Rahouti, Kaiqi Xiong, and Jing Lin. 2021. Leveraging a cloud-based testbed and software-defined networking for cybersecurity and networking education. *Engineering Reports* 3, 10 (2021), e12395. doi:10.1002/eng2.12395

[17] Mohamed Rahouti, Kaiqi Xiong, and Jing Lin. 2023. Developing Blockchain Learning Lab Experiments for Enhancing Cybersecurity Knowledge and Hands-On Skills in the Cloud. In *Computer Science and Education*, Wenxing Hong and Yang Weng (Eds.). Springer Nature Singapore, Singapore, 438–448.

[18] John Rieffel. 2023. *Presentation for mini-symposium on education using Chameleon.* Video available in program listing at: https://chameleoncloud.org/chameleon-cloud-users-meeting/user-meeting-2023/.

[19] Darshan Sarojini and Aroua Gharbi. 2022. IndySCC: A New Student Cluster Competition That Broadens Participation. In *Practice and Experience in Advanced Research Computing 2022: Revolutionary: Computing, Connections, You* (Boston, MA, USA) *(PEARC '22)*. Article 72, 4 pages. doi:10.1145/3491418.3535153

[20] Vicraj Thomas, Niky Riga, Sarah Edwards, Fraida Fund, and Thanasis Korakis. 2016. *GENI in the Classroom.* Springer International Publishing, 433–449. doi:10.1007/978-3-319-33769-2_18

[21] Yongzhi Wang, Wen-Jung Hsin, and Manish Lamsal. 2022. EdGENI: Making GENI User-Friendly for General Computer Education. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1* (Providence, RI, USA) *(SIGCSE 2022)*. Association for Computing Machinery, New York, NY, USA, 801–807. doi:10.1145/3478431.3499300

[22] Alexander Wolosewicz, Prajwal Somendyapanahalli Venkateshmurthy, and Nik Sultana. 2025. Experience Report: Using the FABRIC Testbed to teach a Graduate Computer Networking course. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1* (Pittsburgh, PA, USA) *(SIGCSETS 2025)*. Association for Computing Machinery, New York, NY, USA, 1246–1252. doi:10.1145/3641554.3701923

[23] Yanyan Zhuang, Chris Matthews, Stephen Tredger, Steven Ness, Jesse Short-Gershman, Li Ji, Niko Rebenich, Andrew French, Josh Erickson, Kyliah Clarkson, Yvonne Coady, and Rick McGeer. 2014. Taking a Walk on the Wild Side: Teaching Cloud Computing on Distributed Research Testbeds. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (Atlanta, Georgia, USA) *(SIGCSE '14)*. Association for Computing Machinery, New York, NY, USA, 535–540. doi:10.1145/2538862.2538931

[24] Yanyan Zhuang, Albert Rafetseder, and Justin Cappos. 2013. Experience with Seattle: A Community Platform for Research and Education. In *2013 Second GENI Research and Educational Experiment Workshop*. 37–44. doi:10.1109/GREE.2013.16