# CLaDS: A Cloud-Based Virtual Lab for the Delivery of Scalable Hands-On Assignments for Practical Data Science Education

Chase Geigle
University of Illinois at Urbana-Champaign, USA
geigle1@illinois.edu

Ismini Lourentzou
University of Illinois at Urbana-Champaign, USA
lourent2@illinois.edu

Hari Sundaram
University of Illinois at Urbana-Champaign, USA
hs1@illinois.edu

ChengXiang Zhai
University of Illinois at Urbana-Champaign, USA
czhai@illinois.edu

## ABSTRACT

The rise of the "big data" era has created a pressing demand for educating many data scientists and engineers quickly at low cost. It is essential they learn by working on assignments that involve real world data sets to develop the skills needed to be successful in the workplace. However, enabling instructors to flexibly deliver all kinds of data science assignments using real world data sets to large numbers of learners (both on-campus and off-campus) at low cost is a significant open challenge. To address this emerging challenge generally, we develop and deploy a novel Cloud-based Lab for Data Science (CLaDS) to enable many learners around the world to work on real-world data science problems without having to move or otherwise distribute prohibitively large data sets. Leveraging version control and continuous integration, CLaDS provides a general infrastructure to enable any instructor to conveniently deliver any hands-on data science assignment that uses large real world data sets to as many learners as our cloud-computing infrastructure allows at very low cost. In this paper, we present the design and implementation of CLaDS and discuss our experience with using CLaDS to deploy seven major text data assignments for students in both an on-campus course and an online course to work on for learning about text data retrieval and mining techniques; this shows that CLaDS is a very promising novel general infrastructure for efficiently delivering a wide range of hands-on data science assignments to a large number of learners at very low cost.

## CCS CONCEPTS

• **Social and professional topics** → **Information systems education**; **Information science education**; • **Applied computing** → *Interactive learning environments*; • **Software and its engineering** → *Cloud computing*;

## KEYWORDS

data science education, cloud computing, virtual lab


*ITiCSE'18, July 2–4, 2018, Larnaca, Cyprus*

ACM ISBN 978-1-4503-5707-4/18/07…$15.00
https://doi.org/10.1145/3197091.3197135

**ACM Reference Format:**
Chase Geigle, Ismini Lourentzou, Hari Sundaram, and ChengXiang Zhai. 2018. CLaDS: A Cloud-Based Virtual Lab for the Delivery of Scalable Hands-On Assignments for Practical Data Science Education. In *Proceedings of 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE'18).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3197091.3197135

## 1 INTRODUCTION

Many institutions of higher education have responded to the growing demand within industry for knowledgeable employees in the areas of data science, "big data," and machine learning by providing new interdisciplinary online degree programs in Computer Science that specifically target these areas. These new online degree programs offer undergraduate and postgraduate degrees in an attempt to maximize their impact by allowing for nontraditional students from all over the country (and perhaps the world) to obtain training in these important areas. However, achieving this broad impact through online education comes with challenges that currently lack satisfactory solutions: (1) designing assignments to allow for the development of *hands-on experience* with *real data sets*; (2) deploying those assignments to off-campus students that lack the computing resources traditionally offered to on-campus students; and (3) minimizing the overall cost (both monetary and time) of the deployment of such assignments.

In general, offering large-scale courses targeting a wide variety of students comes with a direct course scalability challenge. How can we ensure that we can deliver meaningful, hands-on experiences to these students that can allow them to develop practical skills? Such practical skills are especially important for an experimental field such as data science. In a traditional classroom setting, we offer programming assignments, but in an online and/or large-scale classroom we must be careful to ensure that our programming assignments can properly scale. While traditional students are often given access to a physical computer lab or shared remote server, it is infeasible in most cases to simply offer online students access to on-campus computing resources, and it is similarly unreasonable to expect all such students to have workstation-level computers capable of handling real-world data sets. Since the real-world data sets cannot be easily moved, using them necessitates the provision of some form of computing to the students on the cloud where the data are stored. However, this must also be done in such a way as to have a minimal impact on tuition costs which already present a large barrier for many students wishing to obtain an accredited

degree in data science. Today, this forces instructors into making a trade-off with their assignments that eschews truly practical experiences (with industry-standard tools and real data sets) for feasibility—students often work on toy problems with tiny data sets in order to better ensure that a student's current computing device can handle the task. This limits a student's ability to learn the skills and tools necessary to be effective in a real industry setting.

To break this bottleneck, we would ideally want to design a system for deploying practical hands-on assignments for data science that satisfies four criteria: (1) it must allow instructors to easily scale their courses to large numbers of students; (2) it must allow for the deployment of a wide spectrum of possible assignment designs to be flexible to handle most, if not all, desired hands-on experience training in data science; (3) it must be able to use real-world datasets to ensure the practicality of the skills students are able to learn; and finally (4) it must do all of these at minimal cost. In this paper, we propose a novel system that meets all of the above criteria called CLaDS: a Cloud-based Lab for Data Science. CLaDS scales to large numbers of students while simultaneously allowing for both the use of real data sets as well as a high degree of assignment design flexibility. We describe the design and implementation of CLaDS, and discuss our practical experiences with deploying CLaDS in two different courses in a data science curriculum. We find that at a cost of as little as $7.40 per student, CLaDS was able to provide a computing environment that facilitated a wide variety of assignments ranging from in-depth analysis of specific algorithms to competition-style assignments in which students approached or beat state-of-the-art solutions to open research problems. Because our proposed virtual lab system is quite general, it holds the promise to pave the way for a more complete data science education when instantiated for a number of different problem domains. Source code and instructions for deploying CLaDS is freely available[1].

## 2 RELATED WORK

Recent literature turns towards producing scalable platforms that can support the full pipeline of programming assignments. For example, Prof. CI [10] allows students to work on their local machines and recieve automated feedback in the form of GitHub issues when their code submissions are pushed to a GitHub repository. Systems like CodeOcean [14], Hackerrank[2], and TopCoder[3] provide web-based platforms that support the execution and assessment of programming exercises. All of these platforms, however, are insufficient for addressing general data science education as they do not support uploading a dataset and thus instructors cannot build assignments that utilize real-world data.

Scaling traditional programming assessments often involves designing and running student code through a battery of unit tests to evaluate correctness [7]. While unit tests can provide partial automation, they do not incorporate instructor rubrics. Data-driven methods have been developed for automatically grading [13] and producing feedback, including leveraging search engines that leverage the redundancy found in highly structured homework [11] and deep learning methods [12]. However, nearly all of such systems focus on programming problems where there is a single "gold standard" solution; in data science there is rarely a single "correct" answer, so facilitating these assignments is difficult through using traditional programming assignment techniques.

A number of general platforms are available for running data science competitions which help data science education. Among them, Kaggle[4] is the closest to ours. It primarily allows for the delivery of data science competitions on labeled datasets, and recently provides an online computing environment that allows users to run scripts (called "kernels") when participating in these competitions. Our system differentiates itself in a few key ways: (1) our system allows complete flexibility in the tools and libraries used in an assignment and customized grading rubrics —Kaggle by comparison has a whitelisted set of libraries and tools that one is allowed to use in a "kernel," (2) our system supports offering traditional assignments that are *not* competitions, and (3) our system does not place any strict limit on the size of the dataset that can be used.

Lopez et al. [8] showed that students working on open-ended challenge problems in machine translation can result in student systems (or combinations of student systems) capable of reaching near state-of-the-art performance. Such a set of assignments is a perfect fit for our virtual lab system, and we should expect to see similar results to theirs across the broad spectrum of *all* applications in data science. There has been some previous work on creating a virtual lab for information retrieval, one subdomain of data science [3, 4]; we generalize this to address creating a virtual lab for *any* data science domain or application.

Our proposed virtual lab makes heavy use of a cloud computing infrastructure; for a comprehensive survey of the use of cloud computing in education, please see González-Martínez et al. [6]. Finally, our system heavily uses the concept of continuous integration (CI) introduced by Beck [1]. CI is a software engineering concept that minimizes the gap between development and production of software; see Fitzgerald and Stol [5] for a comprehensive review of CI in its many forms in software engineering. We build upon the concept of CI by adapting it for use in facilitating running student code on real-world datasets instead of purely for testing software.

## 3 CLADS: A DATA SCIENCE VIRTUAL LAB

A typical data science assignment involves the use of some data set to extract knowledge. This broadly covers areas[5] such as information retrieval (where the goal is to develop a system to respond to queries with relevant data, typically in the form of free-text documents), data mining (where the goal is to directly use the existing data to extract knowledge from statistical patterns present in the data), machine learning (where the goal is to train a model on some data set in order to make predictions about new data), and visualization (where the goal is to create interpretable visual representations of data sets). At a high level, all of these domains involve creating a piece of software that can process a data set and produce some desired output. The usefulness of this output depends on the quality, and in many cases the size of the input data set, and the amount of computational effort required to produce the useful

[1] https://timan-group.github.io/clads/
[2] www.hackerrank.com
[3] www.topcoder.com
[4] www.kaggle.com
[5] This is a representative, but not exhaustive, list of data science subdomains.

output typically scales as a function of the data size (both in terms of the number of items as well as their dimensionality).

Thus, in an ideal setting we would provide students with a real data set to work with, and instruct them on the use of industry-standard tools that have been designed to handle that scale. Unfortunately, in most cases the real data sets that we wish to use are too large to reasonably distribute to students, particularly in an online setting where they would most likely be forced to download it to their own computers. As a result, instructors are generally "forced" to offer assignments that use very small, toy data sets. Unfortunately, observations generated by running algorithms on these very small data sets are known to be misleading, as a small data set often fails to sufficiently capture the true variety present in a real data set. By instead offering the assignment through a cloud-based virtual lab, we can enable students to work on real-world data sets by bypassing the data distribution problem and instead moving student code to where the data resides. Moving student code is cheap—it is easily tens of orders of magnitude smaller than even the smallest of real-world data sets.

## 3.1 Interaction Flow

Our proposed system, CLaDS, solves this problem as follows (see Figure 1 for a detailed graphical overview). At a high level, an assignment delivered through CLaDS has students obtain and submit code to a central authority hosted in the cloud. Upon submitting new code to that central authority, an automated process is invoked that builds and runs that student's new code on a worker machine that is co-located with a real-world dataset within the same cloud infrastructure. Instructors, as well as students, have full control over what tools and libraries they wish to use to process the dataset in order to enable students to gain practical hands-on experience with industry-standard methods. While this code is running, students obtain real-time terminal output from the code through a web-based user interface. When the code finishes running, any output it generates can be saved as an archive that can then be downloaded from that same web user interface (UI), and the worker can then submit results to a leaderboard that can be updated if the assignment has a competition component.

From an instructor perspective, adapting an existing assignment to the virtual lab is relatively straightforward. The data set used for the assignment would first need to be uploaded into the cloud computing infrastructure, and a skeleton for the student code (if desired) would need to be created for distributing to the students through the version control system (VCS); this skeleton code contains a script that is used to install any required tools or libraries, which will be used to configure the worker machines that are eventually used to run the student code. In the event that the assignment has a competition component, code for judging a student assignment submission would need to be uploaded to the leaderboard server, and the skeleton code distributed to the students through the VCS should be updated to include a leaderboard submission script.

## 3.2 Detailed System Architecture

Figure 1 gives a visual overview of our system architecture. Below, we detail the implementation of each of the major components.

**Version Control System.** The first major component of our system is the version control system (VCS) used to store the code. In our case, we use `git` to store a copy of each student's code for a particular assignment. Using a UI for the VCS, such as GitLab[6], provides the students with a central location for their code (as a form of backup) and allows them to work from any location that has an Internet connection. The VCS is the source of "truth" in our system, from which both students and later components of the system obtain the current version of the code for each individual student, assignment pair. Because this system also stores all historical information about all versions of an assignment, it can provide extra information for assessment of student performance and analysis of student learning behavior; furthermore, it can also facilitate reproducible research by serving as an archive of a set of attempted solutions (and their generated results) to an open research problem in data science (which is often completely characterized by a competition challenge with a particular data set).

**Continuous Integration.** The system we propose borrows heavily from the concept of continuous integration (CI) first introduced by Beck [1] in the context of the "extreme programming" software development methodology. CI as a practice has evolved and grown over time to mean different things for different people [5], but the key component of the methodology is to minimize the gap between the development and deployment of software. One typical approach is to tie a software tool that facilitates a build and test cycle on a production (or near-production) environment to VCS being used for the software's development in such a way that every commit to that VCS results in an automated build and test process. Doing this ensures that every (or nearly every) commit to the central repository both builds and runs properly in an environment that is not just the developer's local machine.

The fact that these CI systems build and run the code on a machine that is entirely *separate* from the developer's machine is crucial for our adaptation. The key insight is this: because CI systems move the code onto a separate machine for the build and test process, we can exploit this property to move student code onto a virtual machine in the cloud in the same datacenter that stores some large, real-world data sets. In effect, we "move the code to the data," a common practice in real-world data science applications where it is prohibitively expensive, or in many cases flat out impossible, to move the data set to be analyzed around to different locations.

Running the student code in the cloud has two key benefits. First, it is now possible (perhaps for the first time) to allow students to experiment on data sets that would be impossibly large to deliver in a traditional setting because we are able to run student code on the same cloud infrastructure that is used to store these large real-world data sets. This allows us to greatly reduce the gap between education and practice by allowing students to work on real problems instead of simple (but illustrative) toy examples. Moreover, this separation between working locally and running an analysis on real-world data is consistent with how data scientists work in practice: first making the model work on small, toy data sets, and then testing the model on much larger, real-world data sets of the same format. Second, from an instructor's perspective, this separation allows for a more centralized, predictable computing environment

[6] https://gitlab.com

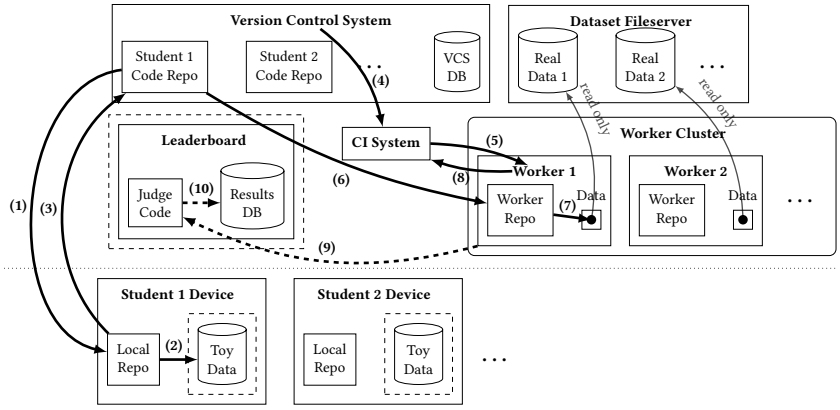


**Figure 1: The overall CLaDS system design. Dashed lines indicate optional components, and the dotted horizontal line indicates the separation between local devices (student machines; bottom) and cloud devices (virtual machines; top). The interaction flow is as follows: (1) student clones/pulls code to local device; (2) student writes code locally (optional: testing on toy data set); (3) student pushes code back to the version control system (VCS); (4) the VCS notifies the continuous integration (CI) system; (5) the CI system spawns a worker (re-using one if available); (6) the worker clones/pulls the student code; (7) the worker builds and runs the student code; (8) the CI system updates its UI as the build/run progresses; (9) *(optional)* the worker submits results to the leaderboard system; (10) *(optional)* the leaderboard system judges the results and updates its competition rankings.**

for each assignment to be offered, freeing teachers from having to design detailed set-up guides and working with individuals on troubleshooting issues for each possible device configuration a student might have, and instead allowing them to focus on teaching the tool itself. Because these systems provide real-time output from the programs in the form of a pseudo-terminal provided in the web UI, the experience difference between running the tools locally vs running them in the cloud is minimized. This allows for the same rapid iteration cycle possible when running code locally.

Another key component of the CI system we used in our virtual lab implementation is its ability to generate "build artifacts" that can be downloaded at the completion of the job. In a traditional software engineering environment, these are typically the compiled binaries or library files generated by a build-and-test process, but in our setting these are often summary reports, generated data tables, graphs and figures, or trained models. In this way, the output generated by student work is not "locked up" in the cloud, but rather is freely available for them to use. Naturally, they can also be used for automated assessment of student work (e.g., by comparing the results produced by the student system with some gold standard).

A number of tools for CI infrastructure exist—in our system, we leveraged the GitLab VCS along with its tightly integrated GitLab CI[7] for this purpose. Both tools are available under liberal open-source licenses.

**Autoscaling.** The GitLab CI software provides an additional feature that completes the scalability picture for our proposed virtual lab adaptation: auto-scaling build workers. In many CI infrastructures, there is a single server, or some fixed set of servers, that are used to process the build-and-test jobs that are created by commits to a software repository. This kind of setup requires careful analysis of the number of machines required to meet build demand. In corporate environments with regular usage, this may be easy to estimate, but in an educational environment the demand is much more variable with high utilization around assignment deadlines with sporadic activity between them. In order to provide a better experience and simultaneously minimize cost, we desire a flexible number of build workers determined by real-time demand.

The auto-scaling feature[8] of GitLab CI works by dynamically spawning new virtual machines in some cloud infrastructure as needed to meet the current demand of build jobs. These spawned machines are kept and re-used to run multiple build jobs until demand lowers and they remain idle for a certain timeout period, after which they are decomissioned. This allows for a perfectly flexible number of build machines to be maintained, and even no machines spawned at all when there are no currently pending jobs.

**Leaderboard.** Open challenge problems can be a powerful tool for learning and they enable learners to explore new ideas for research. To facilitate the delivery of engaging challenge problems, we deployed a web-based leaderboard system as part of our virtual lab. This website contains a list of student results (typically something like accuracy, precision, recall, or other objective metric for the task used in the research literature) along with some baseline(s). Students can then work individually or together in small groups to attempt to devise solutions that beat the baseline (typically a simple, but naïve method) and attempt to beat state-of-the-art models. In practice, we observe students regularly obtaining near state-of-the-art performance, consistent with Lopez et al. [8].

## 4 DEPLOYMENT EXPERIENCE

We deployed a number of assignments using the MeTA text retrieval and mining toolkit [9] through an instantiation of the virtual lab infrastructure in two courses, CS4 and CS5, in the information retrieval and text mining domains, respectively. The delivered assignments (see Table 2) covered a broad spectrum of use-cases for

[7] https://about.gitlab.com/features/gitlab-ci-cd/

[8] https://docs.gitlab.com/runner/install/autoscaling.html

**Table 1: The total number of submissions students made *after beating the baseline submission.* Even in the 25th percentile we see three to five additional submission attempts; 50% of students submitted more than 10 additional attempts.**

| Assignment | Mean | Std. Dev. | Median | 25th %ile |
|---|---|---|---|---|
| CS4-MP2 | 20.5 | 27.6 | 10.0 | 5.0 |
| CS4-MP3 | 21.7 | 42.3 | 10.0 | 3.0 |

the virtual lab, ranging from parameter sensitivity experiments and in-depth algorithm evaluation to competition-style assignments run on real datasets with leaderboards for tracking progress.

Historically, both courses could only use very small "unreal" data sets due to the lack of infrastructure support. With CLaDS, we were able to, for the first time, use much larger real data sets for all the assignments of those two courses, thus enabling students to learn skills that can be directly useful for solving real world problems. While not explored, all the assignments can also be easily made available to any learners around the world that have an Internet connection. Moreover, not only was grading assignments performed automatically with the use of Gitlab API[9], but also the number of configuration problems was decreased compared with previous offerings of the course that relied on implementing assignments locally. Hosting the student code repositories all in one place helped reduce turn-around times for providing guidance to students.

## 4.1 Competition Experience

We offered two competitions through the virtual lab, both in the CS4 course. In both, we maintained a leaderboard that listed each student's current submission score (optionally anonymized) along with a baseline method. Students were required to beat the baseline method to complete the assignment, and a small amount of extra credit incentive was given for being in the top rankings. The first competition (CS4-MP2) was a search engine competition where students competed with one another to improve the relevance of search rankings—the use of the virtual lab here not only simplified the running of the competition component of the assignment, but also allowed us to use a real-world information retrieval dataset that was difficult to use before. The second competition assignment (CS4-MP3) was a geolocation prediction competition [10] using a Twitter dataset that enabled students to compete among themselves and go beyond baselines from existing related work [2]. Our infrastructure setup enabled participants to surpass existing literature, raising the state-of-the-art in the 4-way classification task from 67% accuracy to 75%. Furthermore, several students chose to continue work on this task as part of their course projects, thus the lab serves as a virtual incubator for promoting research in fields of interest.

Student engagement was prevalent throughout the duration of the competition as well as afterwards. We observed that some students experimented with over hundreds of different submissions

[9] https://python-gitlab.readthedocs.io/

[10] The task was to predict the location of twitter users from their textual posts/tweets. All twitter users are from the Contiguous U.S. (i.e., the U.S. excluding Hawaii, Alaska and all off-shore territories) and are classified into 4 classes, which represent the main four U.S. regions (Northeast, Midwest, South and West) as defined by the Census Bureau: https://www2.census.gov/geo/pdfs/maps-data/maps/reference/us_regdiv.pdf.

**Table 2: Topic coverage and total build hours consumed by each individual assignment deployed to the virtual lab in one semester. The total computational resource consumption depends highly on the assignment structure.**

| | CS4 | | CS5 | |
|---|---|---|---|---|
| | Description | Hrs. | Description | Hrs. |
| MP1 | feat. extraction | 28.6 | smoothing methods | 908.8 |
| MP2 | retrieval fns. | 227.1 | word embeddings | 79.3 |
| MP3 | classification | 142.9 | topic mdls. | 250.9 |
| MP4 | - | - | hidden Markov mdls. | 29.6 |

over the course of the assignment period. In Table 1 we show the number of submissions students made to the leaderboard *after* passing the baseline requirement. We can see that students remained engaged with the assignment and its material even after the completion of the main assignment goal; indeed, even the 25th percentile submitted between three to five more times after completing the main assignment, with half of the class submitting 10 or more times post-completion.

In our discussion forum, several students asked for top-ranked submissions to post a description of their solution, and the highest-ranked students responded accordingly continuing the discussion. We also see many cases that expressed positive comments at the end of the course, for example:

- "Professionally speaking, I really feel that I gained a lot, as now I truly understand the essential fundamentals in Text Information Systems areas and, thanks to the hands-on final project, and MPs, can implement some of these principles. I am more than positive that I will utilize the gained knowledge in my workplace in 2018 and make a significant impact."
- "The system used for the programming assignments to automatically test, evaluate, and rank solutions made the assignments a fun challenge."
- "The competition style leader board added a fun aspect..."

## 4.2 Overall System Utilization and Cost

In Figure 2 we denote the total number of running build jobs over time in our virtual lab deployment for the two courses over the whole semester. We can see the expected pattern of most build jobs happening on the eve of deadline periods. In the case where two deadlines were near one another, we saw a peak of over 1,200 build jobs occur in a single day. In between those dates, the build demand is quite variable (and in many cases nonexistent)—this translates to cost savings by leveraging the auto-scaling feature of our CI system to decommission idle machines during such periods, limiting our expenditures to only maintaining the leaderboard server(s), the CI system master machine, and the VCS server in those cases.

Table 2 highlights an important point: the amount of build hours required to run an assignment depends strongly on the assignment design itself. Assignments that are more "light" in nature (e.g. CS4's MP1) do not consume nearly as many build jobs as competition assignments or assignments requiring a significant amount of experimentation work. CS5's MP1 is a good example of an assignment

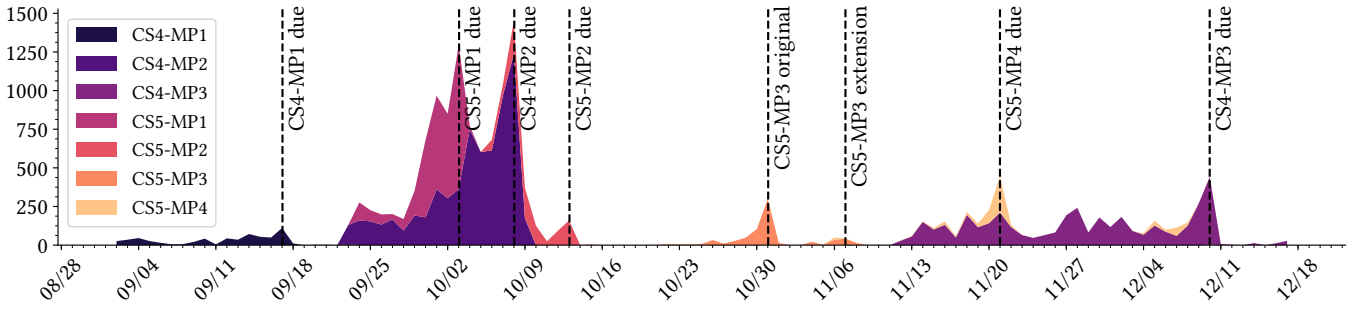


**Figure 2: Total running jobs in the virtual lab over the course of one semester. Data is bucketed by the day, with assignment deadlines denoted with vertical dashed lines. The build demand is highly variable, which allows for significant cost savings when there is little or no demand. We see the highest demand for builds around assignment due dates—in one case where both of our courses had due dates near one another, we saw nearly 1,500 builds performed in a single day.**

that benefits a lot from the virtual lab—using the infrastructure we built, we were able to ask students to experiment on real datasets with a number of different methods and parameters for those methods. This results in a large amount of fairly time consuming build jobs, which our auto-scaling worker cluster handled very well.

The total cost of the deployment of the virtual lab across both the CS4 and CS5 offering simultaneously for one semester was a mere $1680.29 USD which provided access for the 136 students in the CS4 course and the 91 students in the CS5 course. This comes down to approximately $7.40 USD when amortized over all students. We feel this is a very reasonable "lab fee" for sustaining the virtual lab infrastructure, and shows that there is still significant headroom available for using even larger datasets for experiments that may necessitate larger, more expensive worker virtual machines to process for longer periods of time.

## 5 CONCLUSIONS AND FUTURE WORK

We proposed a novel virtual data science lab system, CLaDS, which addresses a serious bottleneck in "big data" education and provides a general solution to the problem of delivering practical hands-on assignments for teaching data science at scale. Through a principled use of an auto-scaling worker cluster running on a cloud computing infrastructure, such a system can allow these practical assignments to be delivered to large numbers of students while simultaneously enabling the use of real-world data sets. We detailed our experience with using CLaDS to offer seven different assignments in two different courses as part of a larger data science curriculum with a cost as low as $7.40 per student. Through this deployment experience, we showed that CLaDS can facilitate the deployment of a wide variety of assignments with low instructor effort, including competitions that enjoyed high student engagement.

As this was the first attempt at using such a novel system, we were relatively conservative with data set size in our first deployment—our results suggest that there is significant headroom for exploring even larger data sets, which we plan to do soon. Also, since CLaDS is a general infrastructure, it can be easily used to host a wide range of assignments in many other data science domains beyond information retrieval and text mining. CLaDS is built entirely with open source software, so it can be used by any instructor of data science in the world to improve and accelerate data science education.

## ACKNOWLEDGMENTS

This material is based upon work supported by the NSF GRFP under Grant Number DGE-1144245, by the NSF Research Program under Grant Number IIS-1629161, by Microsoft Azure, and by a gift fund from Intel under its support program for Big Data Education.

## REFERENCES

[1] Kent Beck. 2000. *Extreme Programming Explained: Embrace Change.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
[2] Jacob Eisenstein, Brendan O'Connor, Noah A Smith, and Eric P Xing. 2010. A latent variable model for geographic lexical variation. In *Proc. EMNLP.* Association for Computational Linguistics, 1277–1287.
[3] Hui Fang, Hao Wu, Peilin Yang, and ChengXiang Zhai. 2014. VIRLab: A Web-based Virtual Lab for Learning and Studying Information Retrieval Models. In *Proc. SIGIR.* ACM, 1249–1250.
[4] Hui Fang and ChengXiang Zhai. 2014. VIRLab: A Platform for Privacy-Preserving Evaluation for Information Retrieval Models. In *Proc. PIR@SIGIR.* 37–38.
[5] Brian Fitzgerald and Klaas-Jan Stol. 2017. Continuous Software Engineering: A Roadmap and Agenda. *Journal of Systems and Software* 123 (2017), 176 – 189.
[6] José A González-Martínez, Miguel L Bote-Lorenzo, Eduardo Gómez-Sánchez, and Rafael Cano-Parra. 2015. Cloud computing and education: A state-of-the-art survey. *Computers & Education* 80 (2015), 132–151.
[7] Mike Joy, Nathan Griffiths, and Russell Boyatt. 2005. The boss online submission and assessment system. *Journal on Educational Resources in Computing (JERIC)* 5, 3 (2005), 2.
[8] Adam Lopez, Matt Post, Chris Callison-Burch, Jonathan Weese, Juri Ganitkevitch, Narges Ahmidi, Olivia Buzek, Leah Hanson, Beenish Jamil, Matthias Lee, Ya-Ting Lin, Henry Pao, Fatima Rivera, Leili Shahriyari, Debu Sinha, Adam Teichert, Stephen Wampler, Michael Weinberger, Daguang Xu, Lin Yang, and Shang Zhao. 2013. Learning to Translate with Products of Novices: A Suite of Open-Ended Challenge Problems for Teaching MT. *TACL* 1 (2013), 165–178.
[9] Sean Massung, Chase Geigle, and ChengXiang Zhai. 2016. MeTA: A Unified Toolkit for Text Retrieval and Analysis. In *Proc. ACL Sys. Demo.* Association for Computational Linguistics, Berlin, Germany, 91–96.
[10] Christoph Matthies, Arian Treffer, and Matthias Uflacker. 2017. Prof. CI: Employing continuous integration services and Github workflows to teach test-driven development. In *2017 IEEE Frontiers in Education Conference (FIE).* IEEE, 1–8.
[11] Andy Nguyen, Christopher Piech, Jonathan Huang, and Leonidas Guibas. 2014. Codewebs: scalable homework search for massive open online programming courses. In *Proc. WWW.* ACM, 491–502.
[12] Chris Piech, Jonathan Huang, Andy Nguyen, Mike Phulsuksombati, Mehran Sahami, and Leonidas Guibas. 2015. Learning program embeddings to propagate feedback on student code. *arXiv preprint arXiv:1505.05969* (2015).
[13] Shashank Srikant and Varun Aggarwal. 2014. A system to grade computer programming skills using machine learning. In *Proc. KDD.* ACM, 1887–1896.
[14] Thomas Staubitz, Hauke Klement, Ralf Teusner, Jan Renz, and Christoph Meinel. 2016. CodeOcean-A versatile platform for practical programming excercises in online environments. In *Global Engineering Education Conference (EDUCON), 2016 IEEE.* IEEE, 314–323.