Virtual Lab Workspace for Programming Computers – Towards Agile STEM Education

Paul Wu Horng-Jyh(✉), Casey How Kiam Cheng, Bryan Lim Yong Tah, Toh Hong Lie, Justin See Tiong Beng, Roy Ong Ban Guan, Jane Tan Jing Yi, Liu Ziwen, Conejos Sheila Maria Arcuino, Luke Peh Lu Chang, and Zhu Yongqing

Singapore University of Social Sciences, Singapore, Singapore
{paulwuhj,caseyhowkc}@suss.edu.sg

Abstract. Lab-based teaching and learning is essential for STEM education as proficiency in the practice of programming is critical; be it for the development or operation of software applications or the provisioning of cloud services. This enables learners to practice conceptual understanding by applying it to solve real-world problems. In the face of rapid advancement and disruptions of technologies, computer lab curriculum needs to be agile to keep pace with ever changing education landscape. In this paper, we describe a case study on the journey taken by the STEM programme of Singapore University of Social Sciences to evolve its lab teaching via developing virtual lab infrastructure and continuously adapting it to teaching needs via the DevOps process. First, lab environments are migrated to cloud-based virtual machines, providing continuous access within or outside of the physical labs. Infrastructure-as-code and automation are applied to continuously develop and to deploy incremental adaptation through containerized apps in a unified workspace. These apps are customized to suit the specific requirements of the lab curriculum, be it for programming, analytics, database management or even cloud computing. Next, special purpose interactive lab guides are developed to provide automatic and interactive feedback to students who are engaged in the lab exercises. Finally, all online activities in the unified workspace are captured for analysis to assess the pedagogical and operational effectiveness of the unified lab workspace. As of date, the virtual lab infrastructure supports around 3500 students in 14 STEM courses annually.

Keywords: Virtual Lab Workspace · Cloud Computing · Agile STEM Education · Authentic Online Learning and DevOps

1 Introduction

The 2019 COVID pandemic had shifted most classes online, and with that a disruption to STEM (Science, Technology, Engineering, and Math) education in many ways. As the pandemic situation improves, more classes are going back to campus. Despite this, online classes continue to be popular with many tertiary institutions incorporating it into a hybrid approach of course delivery. Moving forward, one of the main challenges that many universities face is the problem of providing a cost-effective, yet authentic online lab-based education to their students in this new normal.

In our university, the Virtual Lab Infrastructure (VLI) project aims to leverage cloud technologies to provide students anytime and anywhere access to virtualized lab workspace that persist outside of the constraints of physical labs. The VLI project is motivated to address the needs of all stakeholders, including the students, course instructors, pedagogical researchers and technical support. Although there have been many other VLI initiatives rolled out in institutions worldwide, the novelty of our solution is the simultaneous adoption of both virtual machines (VM) based and container-based technology to enable a flexible and scalable VLI that can on the cloud without any complex setup requirements.

In the remaining part of Sect. 1, we elaborate on the importance of lab-based teaching and learning for students enrolled in STEM education, specifically computer science, the goals of the VLI project, issues encountered in the development and usage of the VLI and the required critical success factors. In Sect. 2, we document our approach to the deployment of cloud computing services and innovation for virtual lab environments. In Sect. 3, we update on the deployment status, issues and solutions of VLI. We conclude our work and share some future initiatives in Sect. 4.

1.1 Importance of Lab-Based Teaching and Learning in STEM Education

For STEM students and especially in computer science, having hands-on practice is essential to the understanding of programming and technical concepts through participation in real-world problem solving in lab environments. [1] has identified that placing learners in genuine professional environments where they can learn, observe, practice and reflect on the skills which they are studying is key to achieving the learning outcomes of the course. In addition, [2] observes that learners that receive coaching and mentoring from the instructor in the professional environment enhance learners' confidence level and capabilities.

Although it can be challenging to replicate the desired experiences mentioned above to a virtual platform, there are several pedagogical approaches that can enhance authentic lab learning online. For example, virtual labs can provide learners with a safe and controlled environment to conduct experiments in. It is also trivial to reset this environment, and allow learners to conduct repeated trials which further enhances their understanding of the programming and technical concept.

1.2 Critical Success Factors for Virtual Lab Platform

The success of any virtual lab platform ultimately depends on three main groups of stakeholders: students, instructors and technical support.

As elaborated in the previous section, hands-on practices provide opportunities for active learning, where students can directly engage with the subject matter, work through real-world problems and apply scientific concepts and engineering principles to find solutions. However, a survey [3] had shown that 66% of college and graduate students expressed that online classes are not as effective as traditional classroom teaching. Another study [4] had also suggested that students in online programs perform worse on nearly all test score measures relative to their counterparts in on-campus programs.

Some of the negative factors which affect user satisfaction were identified in these studies. These include poor accessibility and reliability, lack of support in navigating and using course and lab content, poor collaboration support and so on.

Instructors are responsible for designing the lab exercises and teaching the concepts to students. One of the pressing issues many instructors face is the difficulty in assessing student learning as they may not be able to observe students' work directly. This can make it difficult to evaluate students' mastery of the material and provide feedback that is meaningful and actionable [5].

Technical support plays an important role in providing virtual lab deployment, platform maintenance, user support and addressing concerns related to the usage of the virtual lab platform. Thus, they are a key stakeholder in ensuring the success and sustainability of the virtual lab platform. As virtual labs rely heavily on technology, software and hardware to function effectively, without effective technical support, virtual labs may experience technical issues like crashes, connectivity problems or other errors. If this is not managed properly, these issues can disrupt the learning experience and cause frustration among students. Furthermore, many virtual lab infrastructure requires technical support staff to perform manual deployment and management. This imposes high workload on staff managing the platform and further degrades the quality of support provided to students and instructors.

It is important to address the above issues before the virtual lab platform can become a viable alternative to the traditional physical lab. We identify the following critical success factors for achieving this. Firstly, the virtual lab must be easily accessible and reliable for students to use. It should be highly available with minimal downtime or technical issues. The platform should also be accessible from different devices and operating systems, so that students can use it regardless of their location or device. The requirements strongly suggest the deployment of the online lab on a cloud-based platform.

Secondly, the quality of the lab experience on the virtual lab platform is crucial for its success. For the students, the platform should provide a similar experience to a physical lab, with realistic simulations, experiments, and data analysis. Collaboration and mentoring are an important part of the learning process; thus the platform should provide opportunities for students to interact with lab instructors or peers. It should also have a variety of lab options, covering different topics and skill levels, so that students can be scaffolded into a learning path that align with the cognitive development of programming concepts and skills. For the instructors, the virtual lab platform should also make it easy to keep track of the work done by students through real-time monitoring, data analytics and interactive dashboards. Based on the data, instructors will be able to mentor the students more effectively [6, 7, and 8].

Thirdly, the workload imposed on the technical support staff in the deployment and management of labs on the platform should be minimized. The amount of time and effort that the technical support staff spent on deploying and managing the labs on a virtual platform should be comparable or even less to that of a physical lab.

To achieve the critical success factors as mentioned above, we identify several desirable features the virtual lab infrastructure should possess:

- Remote access and persistent lab environment to provide seamless access in and out of the physical labs for students and instructors

- Facilitating self-assessment and collaboration for students
- Provide access to student learning metrics for instructors or pedagogical researchers
- Adopt automation to reduce the amount of time and effort in deploying and managing labs for technical support staff

2 Deployment of Cloud Computing Services and Innovation for Virtual Lab Environments

The model for delivering the virtual lab environment is given in the following diagram.

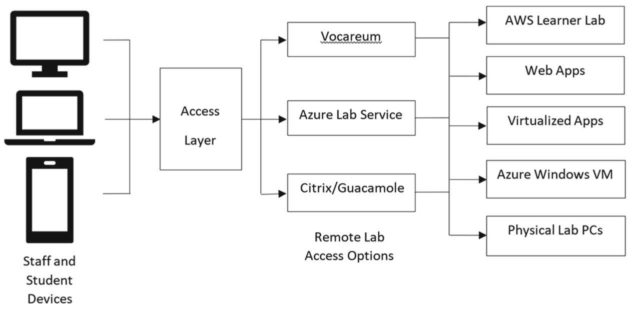


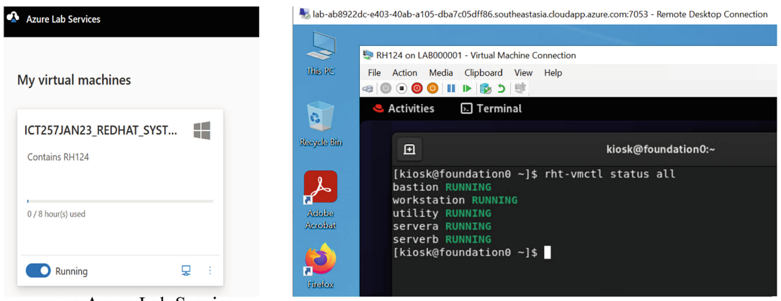
Fig. 1. Virtual lab environment delivery model

2.1 Virtual Machines in the Cloud: Azure Lab Services (ALS)

The first step of the revamping of lab-based teaching is to migrate the lab environments to the cloud. The target courses have rather complex requirements for supporting either nested virtualization environments (Linux virtual machines running inside Windows Virtual Machines) or interface between two application software. ALS-provisioned virtual machines that can meet these requirements. It has also developed an application layer for setting up classroom teaching. Students can be enrolled in a classroom with a template machine that was configured by the lab support team with required tools on the virtual machines. Students access the virtual machines by logging into the ALS portal using their university credentials as shown in Fig. 2 below.

The classroom application layer also allows students' usage of the Lab resources to be managed by a quota system, and when the quota is exhausted the virtual machine will be shut down automatically. We have also set up an application environment through RDP shadowing to allow students and instructors to access the virtual machine concurrently for co-debugging purposes as shown in Fig. 3.

Migrating lab environments to virtual machines is good for replacing physical labs with remote access and on-demand usages, and with interactive co-debugging capabilities, a satisfactory level of interaction between students and instructors is maintained to a similar level to that which is achievable in a physical session in a physical lab.

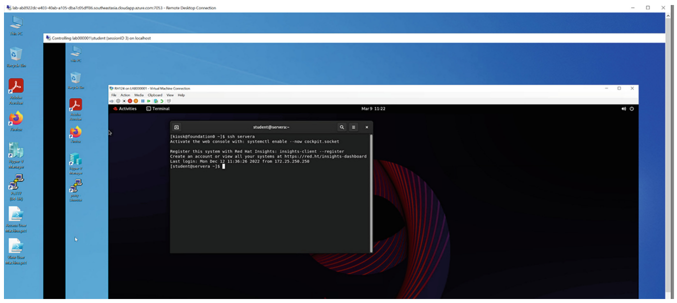

Azure Lab Services
My virtual machines
ICT257JAN23_REDHAT_SYST...
Contains RH124
0 / 8 hour(s) used
Running
lab-ab8922dc-e403-40ab-a105-dba7c05dff86.southeastasia.cloudapp.azure.com:7053 - Remote Desktop Connection
This PC
Recycle Bin
Adobe Acrobat
Firefox
RH124 on LAB000001 - Virtual Machine Connection
File Action Media Clipboard View Help
Activities
Terminal
kiosk@foundation0:~

```
[kiosk@foundation0 ~]$ rht-vmctl status all
bastion RUNNING
workstation RUNNING
utility RUNNING
servera RUNNING
serverb RUNNING
[kiosk@foundation0 ~]$
```

a Azure Lab Service Portal

b Students' VM with Nested Virtualization

Fig. 2. The Azure Lab Services Portal for Students to Access their own Virtual Machines



```
[kiosk@foundation0 ~]$ ssh servera
Activate the web console with: systemctl enable --now cockpit.socket

Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Mon Dec 12 11:36:26 2022 from 172.25.250.250
[student@servera ~]$
```

Fig. 3. Co-debugging session deployed on Azure Lab Services.

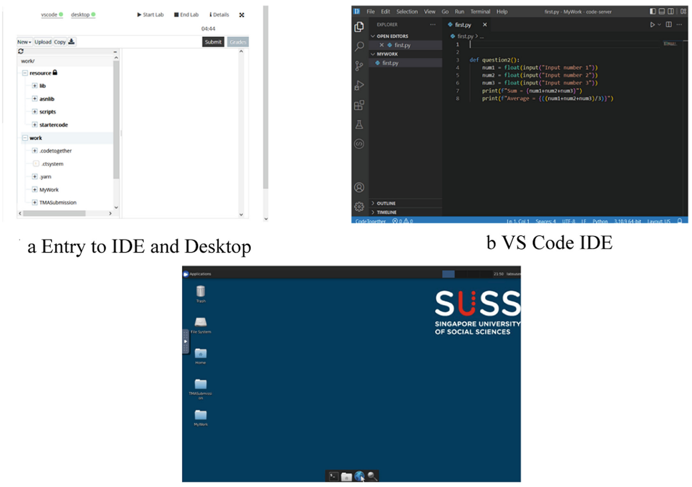
2.2 Containerization of Applications: Vocareum Container Lab

While VMs are a popular way to implement VLI, containers are more agile technologies that provide lightweight virtualization and efficient resource utilization. Furthermore, containers offer a portable and reproducible way of packaging software dependencies and configurations, enabling rapid deployment and orchestration of microservices-based architectures which can be more suitable for Linux based courses.

For our project, the most prevalent container used for lab teaching would be the virtual desktop container. The virtual desktop container provides the GUI to easily perform typical desktop operations, such as creation of folder structures and copying or moving of files as well as running installed applications. Next is the Integrated Development Environment (IDE) container where students use to develop and debug their programming solutions. As shown in Fig. 4, a foundational programming course is being provisioned with VS Code IDE and Linux Virtual Desktop. We also show snippets of docker files that are used to configure the IDE and Virtual Desktop containers on the fly in Fig. 5.

2.3 Cloud Computing as a Service: AWS Learner Lab

In the current age of Cloud computing, lab learning for computer science education would not be complete without the access to Cloud services. AWS Learner Lab is the classroom



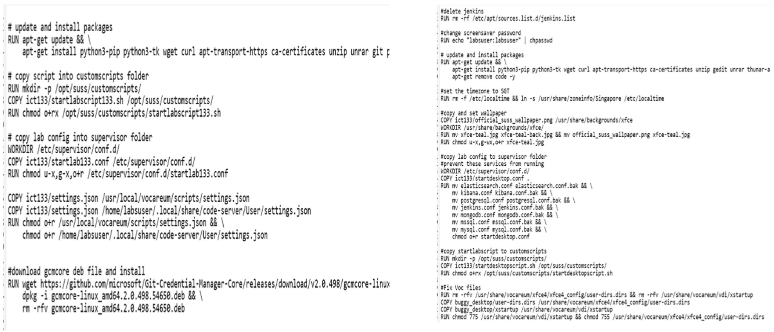
```python

def question2():
    num1 = float(input("Input number 1"))
    num2 = float(input("Input number 2"))
    num3 = float(input("Input number 3"))
    print(f"Sum = {num1+num2+num3}")
    print(f"Average = {((num1+num2+num3)/3)}")
```

a Entry to IDE and Desktop

b VS Code IDE

c Linux Virtual Desktop

Fig. 4. The virtual lab environment that consists of 4.a Entry to IDE and Desktop, 4.b VS Code IDE, and 4.c Linux Virtual Desktop containers



```
# update and install packages
RUN apt-get update && \
    apt-get install python3-pip python3-tk wget curl apt-transport-https ca-certificates unzip unrar git p

# copy script into customscripts folder
RUN mkdir -p /opt/suss/customscripts/
COPY ict133/startlabscript133.sh /opt/suss/customscripts/
RUN chmod o+rx /opt/suss/customscripts/startlabscript133.sh

# copy lab config into supervisor folder
WORKDIR /etc/supervisor/conf.d/
COPY ict133/startlab133.conf /etc/supervisor/conf.d/
RUN chmod u-x,g-x,o+r /etc/supervisor/conf.d/startlab133.conf

COPY ict133/settings.json /usr/local/vocareum/scripts/settings.json
COPY ict133/settings.json /home/labsuser/.local/share/code-server/User/settings.json
RUN chmod o+r /usr/local/vocareum/scripts/settings.json && \
    chmod o+r /home/labsuser/.local/share/code-server/User/settings.json


#download gcmcore deb file and install
RUN wget https://github.com/microsoft/Git-Credential-Manager-Core/releases/download/v2.0.498/gcmcore-linux
    dpkg -i gcmcore-linux_amd64.2.0.498.54650.deb && \
    rm -rfv gcmcore-linux_amd64.2.0.498.54650.deb
```

```
#delete jenkins
RUN rm -rf /etc/apt/sources.list.d/jenkins.list

#change screensaver password
RUN echo "labsuser:labsuser" | chpasswd

# update and install packages
RUN apt-get update && \
    apt-get install python3-pip python3-tk wget curl apt-transport-https ca-certificates unzip gedit unrar thunar-a
    apt-get remove code -y

#set the timezone to SGT
RUN rm -f /etc/localtime && ln -s /usr/share/zoneinfo/Singapore /etc/localtime

#copy and set wallpaper
COPY ict133/official_suss_wallpaper.png /usr/share/backgrounds/xfce
WORKDIR /usr/share/backgrounds/xfce/
RUN mv xfce-teal.jpg xfce-teal-back.jpg && mv official_suss_wallpaper.png xfce-teal.jpg
RUN chmod u-x,g-wx,o+r xfce-teal.jpg

#copy lab config to supervisor folder
#prevent these services from running
WORKDIR /etc/supervisor/conf.d/
COPY ict133/startdesktop.conf .
RUN mv elasticsearch.conf elasticsearch.conf.bak && \
    mv kibana.conf kibana.conf.bak && \
    mv postgresql.conf postgresql.conf.bak && \
    mv jenkins.conf jenkins.conf.bak && \
    mv mongodb.conf mongodb.conf.bak && \
    mv mssql.conf mssql.conf.bak && \
    mv mysql.conf mysql.conf.bak && \
    chmod o+r startdesktop.conf

#copy startlabscript to customscripts
RUN mkdir -p /opt/suss/customscripts/
COPY ict133/startdesktopscript.sh /opt/suss/customscripts/
RUN chmod o+rx /opt/suss/customscripts/startdesktopscript.sh

#Fix Voc files
RUN rm -rfv /usr/share/vocareum/xfce4/xfce4_config/user-dirs.dirs && rm -rfv /usr/share/vocareum/vdi/xstartup
COPY buggy_desktop/user-dirs.dirs /usr/share/vocareum/xfce4/xfce4_config/user-dirs.dirs
COPY buggy_desktop/xstartup /usr/share/vocareum/vdi/xstartup
RUN chmod 775 /usr/share/vocareum/vdi/xstartup && chmod 755 /usr/share/vocareum/xfce4/xfce4_config/user-dirs.dirs
```

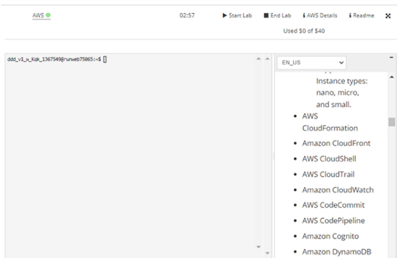
a Docker file for VS Code

b Docker file for Linux Virtual Desktop IDE

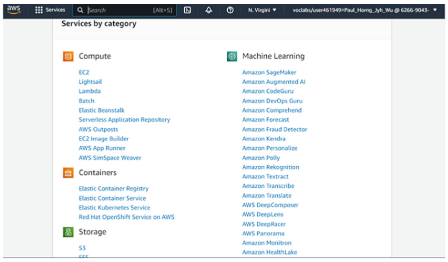
Fig. 5. Docker file that configure IDE and Virtual Desktop containers on the fly

application layer over AWS cloud services. Like ALS, this provides an application layer for quota control over the cloud access that an enrolled student can consume. It also provides a gateway to AWS and for each enrolled student, it provisions a corresponding Identity and Access Management (IAM) to access pre-configured and authorized cloud services. For instance, for an AWS Cloud foundation training course, the provisioned AWS Learner lab will be able to access resources such as EC2, RDS and EBS, and so on many other services (Fig. 6).

However, typical gateway service to cloud services lacks the capability for students to build on their work progressively. The virtual lab environments could support such functions through a virtual machine (VM) disk or a network file system to store away the artifact of students' work. In a typical cloud computing course, the artifact of the students'



a AWS Learner Lab landing page

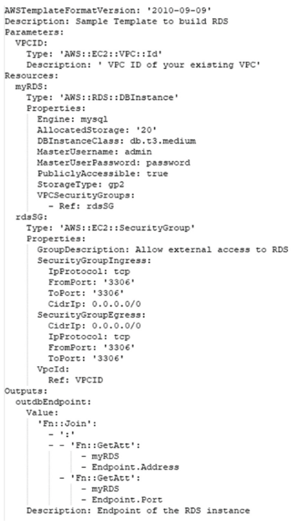


b AWS console for cloud services

Fig. 6. AWS Learner Lab

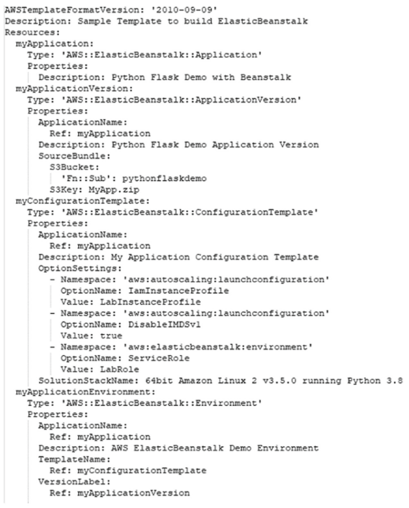
work is that various provisioned cloud services itself, which is typically removed after a lab session by default. For the work to be persistent we need to create an artifact of the students' work to start with. Fortunately, Infrastructure-as-Code affords us to do exactly this. In the case of AWS, CloudFormation allows Cloud architect to program the provisioned cloud service in a JSON- or YAML-document. We introduce such practice to the virtual lab environment for cloud services so that lab work can be persistent through capturing it in a CloudFormation template, we could do the same in other templates such as Terraform as well.

As shown in the following figure, it is an example of the CloudFormation template to provision a RDS and Elastic Beanstalk service (Fig. 7).



```
AWSTemplateFormatVersion: '2010-09-09'
Description: Sample Template to build RDS
Parameters:
  VPCID:
    Type: 'AWS::EC2::VPC::Id'
    Description: ' VPC ID of your existing VPC'
Resources:
  myRDS:
    Type: 'AWS::RDS::DBInstance'
    Properties:
      Engine: mysql
      AllocatedStorage: '20'
      DBInstanceClass: db.t3.medium
      MasterUsername: admin
      MasterUserPassword: password
      PubliclyAccessible: true
      StorageType: gp2
      VPCSecurityGroups:
        - Ref: rdsSG
  rdsSG:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: Allow external access to RDS
      SecurityGroupIngress:
        IpProtocol: tcp
        FromPort: '3306'
        ToPort: '3306'
        CidrIp: 0.0.0.0/0
      SecurityGroupEgress:
        CidrIp: 0.0.0.0/0
        IpProtocol: tcp
        FromPort: '3306'
        ToPort: '3306'
      VpcId:
        Ref: VPCID
Outputs:
  outdbEndpoint:
    Value:
      'Fn::Join':
        - ':'
        - - 'Fn::GetAtt':
              - myRDS
              - Endpoint.Address
          - 'Fn::GetAtt':
              - myRDS
              - Endpoint.Port
    Description: Endpoint of the RDS instance
```

a CloudFormation template for RDS



```
AWSTemplateFormatVersion: '2010-09-09'
Description: Sample Template to build ElasticBeanstalk
Resources:
  myApplication:
    Type: 'AWS::ElasticBeanstalk::Application'
    Properties:
      Description: Python Flask Demo with Beanstalk
  myApplicationVersion:
    Type: 'AWS::ElasticBeanstalk::ApplicationVersion'
    Properties:
      ApplicationName:
        Ref: myApplication
      Description: Python Flask Demo Application Version
      SourceBundle:
        S3Bucket:
          'Fn::Sub': pythonflaskdemo
        S3Key: MyApp.zip
  myConfigurationTemplate:
    Type: 'AWS::ElasticBeanstalk::ConfigurationTemplate'
    Properties:
      ApplicationName:
        Ref: myApplication
      Description: My Application Configuration Template
      OptionSettings:
        - Namespace: 'aws:autoscaling:launchconfiguration'
          OptionName: IamInstanceProfile
          Value: LabInstanceProfile
        - Namespace: 'aws:autoscaling:launchconfiguration'
          OptionName: DisableIMDSv1
          Value: true
        - Namespace: 'aws:elasticbeanstalk:environment'
          OptionName: ServiceRole
          Value: LabRole
      SolutionStackName: 64bit Amazon Linux 2 v3.5.0 running Python 3.8
  myApplicationEnvironment:
    Type: 'AWS::ElasticBeanstalk::Environment'
    Properties:
      ApplicationName:
        Ref: myApplication
      Description: AWS ElasticBeanstalk Demo Environment
      TemplateName:
        Ref: myConfigurationTemplate
      VersionLabel:
        Ref: myApplicationVersion
```

b CloudFormation template for Elastic Beanstalk

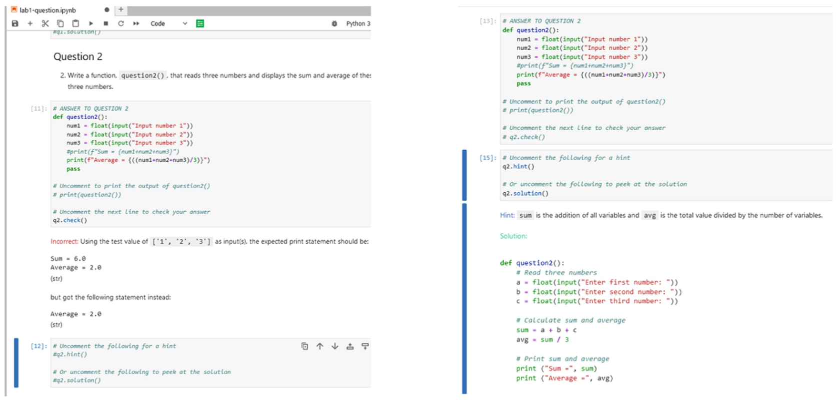
Fig. 7. CloudFormation templates for RDS and Elastic Beanstalk

2.4 Deployment of Interactive Exercise: ILabGuide

For any virtual lab platform to be effective, students need access to authentic assessment for evaluating their knowledge and skills through real-world tasks and activities that simulate the types of challenges they will face in their future careers. In the virtual lab setting, authentic assessment is first achieved by having an interactive lab guide that can auto grade students' submitted solutions.

iLabGuide adopts the learntools framework and packages by Kaggle and further develops customized auto gradable lab exercises according to the design of the instructors. Figure 8 illustrates that for every lab exercise, there is a question statement and a coding area where students can attempt to solve it through multiple iterations. During each iteration, students have the option to deselect certain statements to view the output and verify the accuracy of their solution. If their solution is incorrect, they will receive feedback indicating the input that was incorrect, the expected output, and the actual output obtained.

Figure 8 also shows that after several failed attempts, students may then decide to click on the hints and solution to view the hints and solution as a last resort. In summary, iLabGuide trains students' ability to self-study and practice their skills as a test to authenticate their understanding of the subject matter. Instructors can also design meaningful lab exercises to guide students in a learning path that builds on proper prior understanding of the subject matter. For technical support, iLabGuide is just another container app that can be agilely developed and deployed through the container lab architecture as shown in Fig. 9.



```
#q1.solution()
```

Question 2

2. Write a function, `question2()`, that reads three numbers and displays the sum and average of thes three numbers.

```
[11]: # ANSWER TO QUESTION 2
      def question2():
          num1 = float(input("Input number 1"))
          num2 = float(input("Input number 2"))
          num3 = float(input("Input number 3"))
          #print(f"Sum = {num1+num2+num3}")
          print(f"Average = {((num1+num2+num3)/3)}")
          pass

      # Uncomment to print the output of question2()
      # print(question2())

      # Uncomment the next line to check your answer
      q2.check()
```

Incorrect: Using the test value of `['1', '2', '3']` as input(s), the expected print statement should be:

```
Sum = 6.0
Average = 2.0
```
(str)

but got the following statement instead:

```
Average = 2.0
```
(str)

```
[12]: # Uncomment the following for a hint
      #q2.hint()

      # Or uncomment the following to peek at the solution
      #q2.solution()
```

```
[13]: # ANSWER TO QUESTION 2
      def question2():
          num1 = float(input("Input number 1"))
          num2 = float(input("Input number 2"))
          num3 = float(input("Input number 3"))
          #print(f"Sum = {num1+num2+num3}")
          print(f"Average = {((num1+num2+num3)/3)}")
          pass

      # Uncomment to print the output of question2()
      # print(question2())

      # Uncomment the next line to check your answer
      # q2.check()
```

```
[15]: # Uncomment the following for a hint
      q2.hint()

      # Or uncomment the following to peek at the solution
      q2.solution()
```

Hint: `sum` is the addition of all variables and `avg` is the total value divided by the number of variables.

Solution:

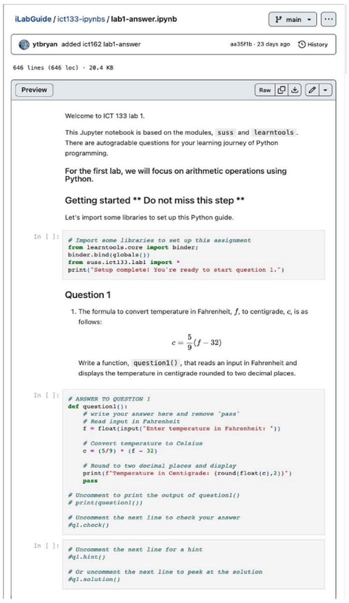
```
def question2():
    # Read three numbers
    a = float(input("Enter first number: "))
    b = float(input("Enter second number: "))
    c = float(input("Enter third number: "))

    # Calculate sum and average
    sum = a + b + c
    avg = sum / 3

    # Print sum and average
    print ("Sum =", sum)
    print ("Average =", avg)
```

a Incorrect attempt with feedback

b Hints and solution as last resorts

Fig. 8. Interactive Lab exercises of using iLabGuide

Last but not the least, iLabGuide allows pedagogical researchers to collect progressive learning activity data that can be analyzed to better inform curriculum designers on issues that may be addressed. Learning activity data consists not only in terms of quantitative information on how many times, success or failures that students have attempted an exercise to demonstrate their understanding of a certain topic, but also qualitative data



iLabGuide / ict133-ipynbs / lab1-answer.ipynb

ytbryan added ict162 lab1-answer aa35f1b · 23 days ago History

646 lines (646 loc) · 20.4 KB

Welcome to ICT 133 lab 1.

This Jupyter notebook is based on the modules, `suss` and `learntools`. There are autogradable questions for your learning journey of Python programming.

For the first lab, we will focus on arithmetic operations using Python.

Getting started ** Do not miss this step **

Let's import some libraries to set up this Python guide.

```
In [ ]:
# Import some libraries to set up this assignment
from learntools.core import binder;
binder.bind(globals())
from suss.ict133.lab1 import *
print("Setup complete! You're ready to start question 1.")
```

Question 1

1. The formula to convert temperature in Fahrenheit, f, to centigrade, c, is as follows:

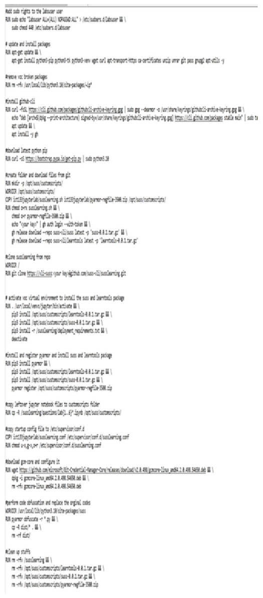
$$c = \frac{5}{9}(f - 32)$$

Write a function, `question1()`, that reads an input in Fahrenheit and displays the temperature in centigrade rounded to two decimal places.

```
In [ ]:
# ANSWER TO QUESTION 1
def question1():
    # write your answer here and remove `pass`
    # Read input in Fahrenheit
    f = float(input("Enter temperature in Fahrenheit: "))

    # Convert temperature to Celsius
    c = (5/9) * (f - 32)

    # Round to two decimal places and display
    print(f"Temperature in Centigrade: {round(float(c),2)}")
    pass

# Uncomment to print the output of question1()
# print(question1())

# Uncomment the next line to check your answer
#q1.check()
```

```
In [ ]:
# Uncomment the next line for a hint
#q1.hint()

# Or uncomment the next line to peek at the solution
#q1.solution()
```

a GitHub Repository for a programming lab exercise



```
#add sudo rights to the labsuser user
RUN sudo echo "labsuser ALL=(ALL) NOPASSWD:ALL" > /etc/sudoers.d/labsuser && \
    sudo chmod 440 /etc/sudoers.d/labsuser

# update and install packages
RUN apt-get update && \
    apt-get install python3-pip python3-tk python3-venv wget curl apt-transport-https ca-certificates unzip unrar git pass gnupg2 apt-utils -y

#remove voc broken packages
RUN rm -rfv /usr/local/lib/python3.10/site-packages/~ip*

#install github-cli
RUN curl -fsSL https://cli.github.com/packages/githubcli-archive-keyring.gpg | sudo gpg --dearmor -o /usr/share/keyrings/githubcli-archive-keyring.gpg && \
    echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/githubcli-archive-keyring.gpg] https://cli.github.com/packages stable main" | sudo te
    apt update && \
    apt install -y gh

#download latest python pip
RUN curl -sS https://bootstrap.pypa.io/get-pip.py | sudo python3.10

#create folder and download files from git
RUN mkdir -p /opt/suss/customscripts/
WORKDIR /opt/suss/customscripts/
COPY ict133jupyterlab/susslearning.sh ict133jupyterlab/pyarmor-regfile-3500.zip /opt/suss/customscripts/
RUN chmod o+rx susslearning.sh && \
    chmod o+r pyarmor-regfile-3500.zip && \
    echo "<your key>" | gh auth login --with-token && \
    gh release download --repo suss-vli/suss latest -p 'suss-0.0.1.tar.gz' && \
    gh release download --repo suss-vli/learntools latest -p 'learntools-0.0.1.tar.gz'

#clone susslearning from repo
WORKDIR /
RUN git clone https://vli-suss:<your key>@github.com/suss-vli/susslearning.git


# activate voc virtual environment to install the suss and learntools package
RUN . /usr/local/venvs/jupyter/bin/activate && \
    pip3 install /opt/suss/customscripts/learntools-0.0.1.tar.gz && \
    pip3 install /opt/suss/customscripts/suss-0.0.1.tar.gz && \
    pip3 install -r /susslearning/deployment_requirements.txt && \
    deactivate

#install and register pyarmor and install suss and learntools package
RUN pip3 install pyarmor && \
    pip3 install /opt/suss/customscripts/learntools-0.0.1.tar.gz && \
    pip3 install /opt/suss/customscripts/suss-0.0.1.tar.gz && \
    pyarmor register /opt/suss/customscripts/pyarmor-regfile-3500.zip

#copy leftover jupyter notebook files to customscripts folder
RUN cp -R /susslearning/questions/lab{1..6}*.ipynb /opt/suss/customscripts/

#copy startup config file to /etc/supervisor/conf.d
COPY ict133jupyterlab/susslearning.conf /etc/supervisor/conf.d/susslearning.conf
RUN chmod u-x,g-x,o+r /etc/supervisor/conf.d/susslearning.conf

#download gcm-core and configure it
RUN wget https://github.com/microsoft/Git-Credential-Manager-Core/releases/download/v2.0.498/gcmcore-linux_amd64.2.0.498.54650.deb && \
    dpkg -i gcmcore-linux_amd64.2.0.498.54650.deb && \
    rm -rfv gcmcore-linux_amd64.2.0.498.54650.deb

#perform code obfuscation and replace the orginal codes
WORKDIR /usr/local/lib/python3.10/site-packages/suss
RUN pyarmor obfuscate -r *.py && \
    cp -R dist/* . && \
    rm -rf dist/

#clean up stuffs
RUN rm -rfv /susslearning && \
    rm -rfv /opt/suss/customscripts/learntools-0.0.1.tar.gz && \
    rm -rfv /opt/suss/customscripts/suss-0.0.1.tar.gz && \
    rm -rfv /opt/suss/customscripts/pyarmor-regfile-3500.zip
```

b Corresponding docker file for deployment into iLabGuide workspace

Fig. 9. Source code for developing and deploying iLabGuide

of the source codes that have been submitted in the attempts. Through machine learning techniques, the content of the source codes could be mined to detect common programming errors or to predict positive or negative direction of progress that the students are making in their attempts. With the recent progress of AI chatbots like ChatGPT, there is even a possibility for iLabGuide to provide an artificial paired programmer that students could co-develop their solution with virtually.

3 Deployment Status, Issues and Solutions

3.1 Deployment Status

Table 1 shows the list of courses that are running on VLI as of December 2022. Information such as course title, tools used and the type of solution adopted is also provided. Due to differing course requirements and platform capability limitations, there are a multitude of platforms involved depending on specifics such as support for containers, VMs and OSes. This could lead to an issue known as platform fragmentation which we will further elaborate on and address in the future work section.

3.2 Managing Across Various Virtualization Solutions Using DevOps Tools

Establishing the DevOps practices within the VLI team itself is critical, as there is typically a lab environment design and deploy cycle followed by quick bug fixing cycles given

Table 1. Course title, tools used and solution type.

Course Title	Tools used	Platform
Structured Programming	VS Code, Jupyter Lab	Vocareum
Computer Architecture	Easy68K, Wine	Vocareum
Object Oriented Programming	VS Code, Python	Vocareum
Data Programming	VS Code, Jupyter Lab, MongoDB Compass	Vocareum
Operating Systems	Virtualbox with extension	Azure Lab Services
Red Hat System Administration	Red Hat Enterprise Linux System	Azure Lab Services
Cloud Computing: Business Case and Technical Models	Amazon EC2, Amazon Elastic Beanstalk	AWS Learner Labs
Big Data Computing in the Cloud	Amazon EMR	AWS Learner Labs
Web Application Development	VS Code, Code-Server, Flask, MongoDB Compass	Vocareum
Oracle Certified Associate	Oracle Database System, JAVA JDK, SQL Developer	AWS
Computer Networking	Cisco Packet Tracer	Vocareum
Database Management systems	SQL Server Express, Azure Data Studio, SQL Management Studio, ER Assistant	Azure Lab Services
Building Information Modeling for Facilities Management	Autodesk Revit, Archibus	Azure Lab Services
Information Security Offence Defence and Incident Management	Cacti, ELK, Nessus, OpenVAS	Azure Lab Services

there are many integration points in the multiple solutions adopted by the team. Thus, the VLI team adopts a suite of tools to support such DevOps practice over successive improved versions of the virtual lab environments.

New source code development is captured in a GitHub repository where the development and testing efforts can be coordinated among the team. Once a particular feature is ready to be rolled out, CI/CD processes will be kicked in to automate the process as much as possible. For instance, for Vocareum container lab, a continuous integration and continuous delivery (CI/CD) process can be implemented by starting a container building process using Dockerfile that in terms starts with cloning new development from Github directly.

Once the building process is completed, a new lab environment would have been deployed when students next try to access the lab environment. Similarly, CloudFormation templates for AWS VPC, EC2, RDS, Elastic Beanstalk and CodePipeline can deploy new applications continuously when GitHub update is detected by CodePipeline. There are other Infrastructure-as-Code (IaC) and automation tools to achieve CI/CD, such as those provided by GitHub Action and Ansible.

Once the application is developed and deployed, operational issues are tracked by MS-Teams message channels as well as VLI support email. With recurrent issues, improvement to the design will be initiated as stories in the Jira system, or new application ideas may be conceived to value-add to virtual lab environments as an Epics. Each story is systematically disposed into tasks to be performed and when it is ready resources will be deployed and a Sprint created to monitor the progress of the development with GitHub as source code control as discussed in the earlier sections.

3.3 Technology Adoption Issues and Resolutions

According to a survey conducted for the Red Hat system administration course, nearly 85% of students spend more than 20 h on lab time. On average, students spend approximately 3.3 h per week on lab time. This represents a greater than 50% increase in lab time compared to a traditional physical lab (Fig. 10).

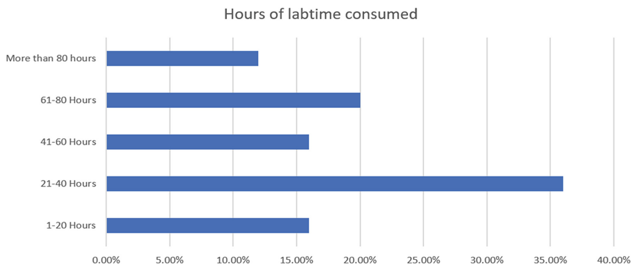


Fig. 10. Survey results of hours of lab time consumed

Over 75% of students gave positive feedback (rating 4 or 5) on the efficacy of the virtual lab, where a rating of 1 denotes highly ineffective and a rating of 5 denotes highly effective (Fig. 11).

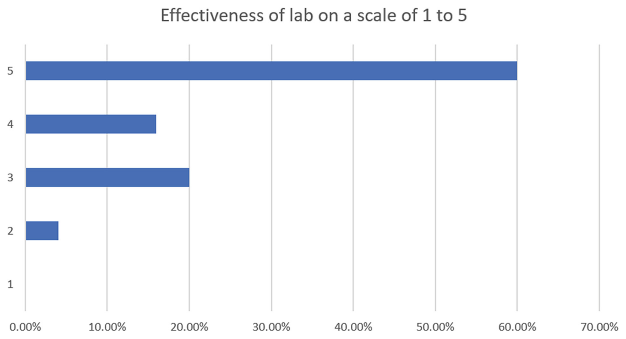


Fig. 11. Survey results of lab effectiveness

We conducted a survey to gather students' opinions on various aspects of the virtual lab, including their likes and dislikes. Many students appreciated the lab's accessibility, as it eliminated the need for physical presence. Additionally, some students found it convenient that they were not limited to specific times or devices for lab access. However, some students reported issues with the deployment of the lab. Specifically, some students found the virtual lab platform to be sluggish during peak hours, while others experienced sudden disconnection problems. This feedback suggests that improvements are needed to enhance the performance and reliability of the virtual lab platform.

The "Building Information Modelling (BIM) for Facility Management (FM)" course has also integrated the VLI platform, reflecting a growing trend accelerated by the global pandemic. The course leverages the VLI platform for lab sessions, aiming to equip undergraduate students with the necessary skills to navigate BIM and BIM for FM software, as well as impart comprehensive knowledge of the best practices and techniques in BIM that can enhance facilities management, including space management.

According to feedback from students and instructors, the adoption of the VLI platform for the "BIM for FM" course has been accompanied by several challenges. One issue that emerged was latency due to the high requirements of the BIM software and the VM specifications. However, this problem was resolved by subscribing to high-configuration VMs.

Another challenge was the dual monitoring requirement during lab sessions, where students needed to switch frequently between the instructor's screen and the VLI interface. To address this inconvenience, it is recommended that students use a second screen or a tablet to observe the instructor's screen and a computer to complete the lab exercises. Additionally, a significant amount of time was devoted to familiarizing students with the VLI interface and functions as many enrolled students had limited IT skills. To overcome this issue, the VLI guidelines were provided to help students better navigate the platform.

While the implementation of VLI in the course "BIM for FM" has proven to be an effective solution for facilitating remote teaching and learning, it has brought certain challenges that should be addressed to optimize the learning experience for BE students.

From what we can observe so far, challenges related to the adoption of technology in virtual lab environments can present obstacles for both institutions and students that can impact negatively on the effectiveness of virtual lab environments and, consequently, on the overall learning experience. We proposed several strategies to deal with these challenges. Firstly, institutions can provide technical support to students to ensure that they can access and use the virtual lab environment effectively. This can include providing tutorials, troubleshooting guides, and access to help desks or online forums.

Secondly, faculty and staff members can be trained to effectively use the virtual lab environment and provide support to students. This can include training on the use of the virtual lab environment, best practices for facilitating virtual lab sessions, and how to troubleshoot common technical issues.

Thirdly, encourage collaboration among students, faculty, and researchers using virtual lab environments. This helps to create a sense of community and support that can help to overcome technological adoption issues.

Finally, institutions can provide feedback to students on their performance in virtual lab environments. This can include feedback on their experimental technique, data collection, and analysis, as well as feedback on their use of the virtual lab environment itself.

Providing feedback can help to improve the effectiveness of virtual lab environments and the overall learning experience.

4 Conclusion and Future Work

In this paper, we have summarized the effort of the VLI project in our university to provide students with access to virtualized lab environments anytime and anywhere. As an ongoing project, the VLI has plenty of room for improvement. We specifically narrow it down to two categories: platform fragmentation and new features.

As seen in Fig. 1, differing course requirements to support containers, VMs, different OSes and so on can result in students having to navigate multiple systems with different interfaces, procedures, and requirements. This is known as platform fragmentation and which can lead to unnecessary cognitive load, reduced engagement, and limit the ability of students to focus on the core concepts and skills they need to master. Platform fragmentation also makes operational support unnecessarily complex and difficult to scale.

To resolve this, we plan to migrate to a single unified platform that supports both containers and VMs with different OS support. It should also have roaming profile support to maintain users' personalized settings, preferences and data across different devices and sessions. Finally, the unified platform should support a distributed file system to provide the storage, availability, performance, and security features necessary to support the complex and dynamic data requirements of virtual lab environments.

To provide more value-added services for learning support, we also identified additional features to be developed:

- Collection and analysis of online activity data for learning analytics [8]
- DevOps of value-added services such as assignment submitter
- Integrating student portfolio for computational competency via GitHub and GitHub Classroom

To conclude, our primary goal for the VLI project is to develop a high-quality and cost-effective platform for laboratory experiments, which offers a fully immersive and accessible experience for its users. In the long term, we hope to make a significant contribution to scientific research and education in our university through this project by engaging all stakeholders, including students, instructors, lab tech team and pedagogy researchers.

References

1. Martínez-Argüelles, MJ., Plana-Erta, D., Fitó-Bertran, À.: Impact of using authentic online learning environments on students' perceived employability. Educ. Tech. Res. Dev. (2022). https://doi.org/10.1007/s11423-022-10171-3
2. Gamage, K.A., Perera, D.A., Wijewardena, M.A.: Mentoring and coaching as a learning technique in higher education: the impact of learning context on student engagement in online learning. Educ. Sci. (2021)

3. Impact of Coronavirus on Students' Academic Progress and College Plans. https://www.niche.com/about/enrollment-insights/impact-of-coronavirus-on-students-academic-progress-and-college-plans#college
4. Cellini, S.R., Grueso, H.: Student learning in online college programs. AERA Open **7** (2021). https://doi.org/10.1177/23328584211008105
5. Guangul, F.M., Suhail, A.H., Khalit, M.I., Khidhir, B.A.: Challenges of remote assessment in higher education in the context of COVID-19: a case study of Middle East College. Educ. Assess. Eval. Account. **32**(4), 519–535 (2020). https://doi.org/10.1007/s11092-020-09340-w
6. Mead, J., et al.: A cognitive approach to identifying measurable milestones for programming skill acquisition. ACM SIGCSE Bull. **38**(4), 182–194 (2006)
7. Xie, B., et al.: A theory of instruction for introductory programming skills. Comput. Sci. Educ. **29**(2–3), 205–253 (2019)
8. Rao, D.M.: Experiences with auto-grading in a systems course. In: 2019 IEEE Frontiers in Education Conference (FIE), pp. 1–8. IEEE, October 2019