World Transactions on Engineering and Technology Education *© 2018 WIETE*
Vol.16, No.2, 2018

Teaching operating systems concepts using the cloud

Mohammed A. Gaffar & Hassan Hajjdiab

Abu Dhabi University
Abu Dhabi, United Arab Emirates

ABSTRACT: Everyone has probably taken some courses on-line, either on YouTube or some other learning platform. Because of cloud computing, it is now possible for remote students to conduct practical assignments on the cloud and to have the same experience as regular students taking courses at their local universities. The cloud platform provides scalable, on-demand and customisable virtual machines for the teachers and students. Cloud-based laboratories have advantages over traditional laboratories since they allow the laboratory resources to expand or shrink based on the number of the students. Reported in this article is an approach to teaching local and remote students a course on operating systems concepts using Linux on the cloud. The key functionality allows the instructor to manage students' accounts and their enrolments. Also post-cloud based laboratory exercises help the student acquire operating systems concepts and skills, such as managing and running processes, threads, pipes and sockets. The cloud services platform used was Amazon AWS (Amazon Web Services); but any other cloud platform will have a similar architecture and could be used.

INTRODUCTION

Cloud computing has evolved rapidly, creating previously unimaginable and unlimited application opportunities. Many individuals and entities including governments, corporations, and even entire industries are now moving to use cloud services to supply their computational and processing needs [1].

One such entity is the educational sector, where many individuals and universities have started to make use of cloud services for their teaching courses. Beyond presenting video lectures, cloud computing allowed education providers to create virtual practice laboratory environments for their students (consumers). There are many advantages to having a virtual practice laboratory environment on the cloud, including but not limited to:

1) having a scalable and on-demand number of virtual machines for students, which eventually cuts down the costs to the education providers;
2) allowing remote students to have the same practical experience as the students taking the course in the physical local laboratories;
3) minimising the computer resource requirements for the students: any average computer with Internet connection will allow the students to perform their tasks since the resources on the cloud are scalable, up or down, based on the needs of the education providers.

> *An operating system is the most important software that runs on a computer. It manages the computer's memory and processes, as well as all of its software and hardware. It also allows communication with the computer without knowing how to speak the computer's language. Without an operating system, a computer is useless* [2].

In this work, the authors focused on teaching operating system concepts using a Linux operating system platform. Linux was chosen because it is open source, most servers run Linux as the operating system, and it is relatively easy to customise compared to other operating systems, such as Windows and MacOS [2].

Why is studying operating systems important? The simplest answer is concurrency. Programming students will have to deal with concurrency somewhere in their careers, if not from the beginning. Operating systems courses are usually the first course that introduces students to threads, processes, sockets, pipes, etc [3].

Concurrent coding concepts are similar for operating systems as for elsewhere in computing [3]. Another important reason to learn about operating systems is to acquire knowledge about client-server concepts and communications.

There are many concepts of operating systems that can be taught on the cloud, such as processes, threads, deadlocks, pipes and sockets. In this article, sockets will be taught using the C programming language on Ubuntu, which is an open source software operating system that is a Linux distribution on Amazon AWS (Amazon Web Services).

Student Socket Assignment

Sockets allow the sending and receiving of data between two processes running on the same machine or on different machines. Sockets mostly are used for client-server communications. In the client-server model, the client knows where the server is and how to connect to it, while the server does not know about the clients; rather it waits for the clients to connect to it. There are two essential types of socket; stream and datagram. Stream is based on TCP (transmission control protocol), which is reliable and connection-oriented, while datagram is based on UDP (user datagram protocol) that is not as reliable and is connectionless.

The socket assignment for the students will be organised as follows: it will have a server socket running during the assignment and the students will be informed of the server port. The students must write a client socket that will connect to the server and read a random sequence generated by the server and save it in a file named with the student's name. When the client connects to the server, the server will generate a random sequence and save it in a file for later verification. At the end of the assignment session, the instructor will run a script that will iterate through all the student's directories reading what the client has saved in the file and see if it matches one of the sequences in the server file; if there is a match, the student will be graded accordingly.

RELATED WORK

In earlier versions of remote laboratories, before the cloud, remote access to a laboratory was given to students, which allowed them to use and manage the available physical machines in the laboratory [4]. Similar to the solution proposed here, Salah et al proposed a cyber-security laboratory on the cloud that is on-demand, and scalable with easily configurable machines for an unlimited number of students [5]. The virtual laboratory allowed students to remotely create virtual machines and conduct cyber-security experiments on them [6][7]. In this article, it is proposed to use a single virtual machine to conduct operating system concepts experiments. This saves costs for the organisation that does not need a physical laboratory and allows the instructor to better optimise the experiments.

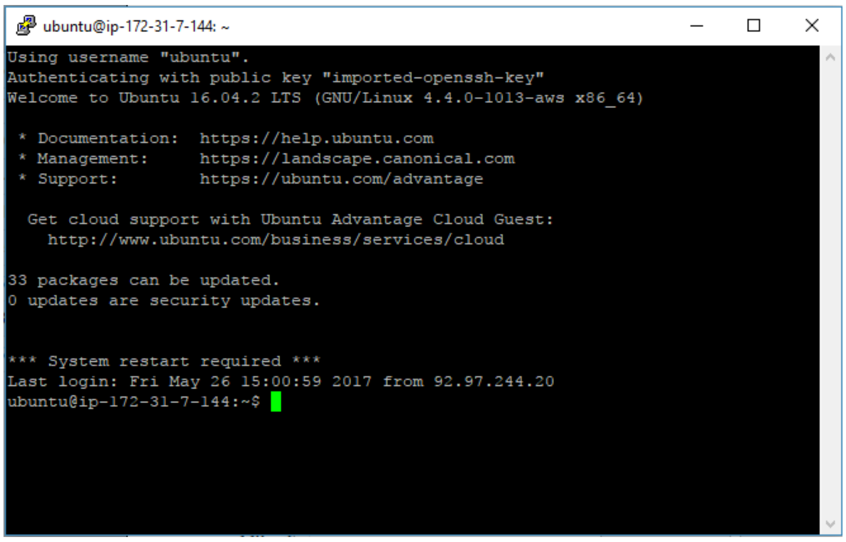
METHODOLOGY

In this section, a detailed description will be given of how to create the operating system sockets assignment. Some of the commands can be combined into a shell script, for demonstration purposes. Such a shell script is not detailed in this article.

Create an Ubuntu Instance on AWS and Connect to it

After registering with Amazon AWS, go to the Amazon AWS dashboard, and click on Launch a virtual machine and choose Ubuntu; then proceed with clicking next until your instance is ready and running. Then, click on your Ubuntu instance and click on connect and follow the instructions to connect to your Ubuntu instance on the cloud (see Figure 1).

Setting up the Linux Environment



```
Using username "ubuntu".
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-1013-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

33 packages can be updated.
0 updates are security updates.


*** System restart required ***
Last login: Fri May 26 15:00:59 2017 from 92.97.244.20
ubuntu@ip-172-31-7-144:~$
```

Figure 1: SSH (secure socket shell) client terminal of the Ubuntu instance.

After logging in to the Ubuntu instance on the cloud, create a group called students by executing the following command:

```
$ sudo groupadd students
```

Create students using the following command. After running the command, it will ask for a password - the default password is “adu123”. Students will be asked to change their password when they login.

```
$ sudo adduser “student”
```

Then, one adds the student to the group:

```
$ sudo usermod -a -G students “student”
```

To give the student the ability to SSH into his account add a public key to the student directory, so the student can connect using the private key. To do that, run these commands:

```
$ cd /home/”student”
$ sudo mkdir .ssh
$ chmod 700 .ssh
$ touch .ssh/authorized_keys
$ cp /home/ubuntu/authorized_keys .ssh
$ chmod 600 .ssh/authorized_keys
$ chown “student”:students .ssh –R
```

It is very important to give the access code of 600 to the public key of SSH, so only the current user can read it. After applying the steps above the student can now SSH from any computer to his username space on the cloud.

Setting up the Server Socket

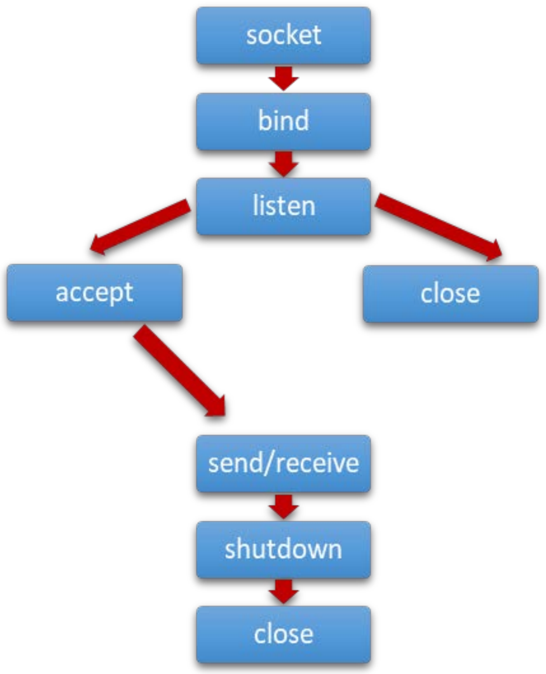


Figure 2: TCP socket server phases.

Figure 2 shows the phases to create a TCP socket server. The first phase is to create the socket. To create the socket, a socket descriptor is needed, which is an integer and works like a file handler. The code below shows how to create the socket descriptor. It only creates the interface without specifying where the data will be coming from or going to:

```
int socket_fd =
socket(AF_INET, SOCK_STREAM, 0);
```

where AF_INET is the communication domain and it associates a socket with the Internet protocol family,
SOCK_STREAM communication type,
0 is the protocol, which is usually set to 0 [5].

Next is the bind phase. In this phase, a socket is bound to an address and a port. The code below is to bind the socket [5]:

```
struct sockaddr_in srv;
srv.sin_family = AF_INET;
srv.sin_port = htons(80);
srv.sin_addr.s_addr = htonl(INADDR_ANY);
bind(socket_fd, (struct sockaddr*) &srv, sizeof(srv));
```

The bind function takes the following arguments [5][8]:

- socket_fd: the socket file descriptor
- &srv: points to a sockaddr structure that contains the address that will be bound to the socket.
- sizeof(srv): specifies the length of the sockaddr structure above.

The server is not ready yet to communicate with a client. The next phase is to listen. In this phase, the server will listen to client connection requests and accept them. In the listen phase, there will be a queue of pending client connections. The code below shows the listen function:

```
listen(socket_fd, backlog);
```

The listen function takes the following arguments:

- socket_fd: the socket file descriptor
- backlog: no of connections that can wait in listen queue of the socket.

However, the server is still not quite ready to communicate with a client. The next phase is the accept phase. In this phase, the accept function extracts the first connection from the pending connections queue in the listen phase and assigns a new socket descriptor and socket type protocol and address family with the same properties of the above socket_fd. Here is the accept function code:

```
newfd = accept(fd, (struct sockaddr*) &cli, (socklen_t*) &cli_len);
```

The accept function returns a new socket with the same properties as the original socket_fd, and it takes the following arguments [5]:

- socket_fd: socket descriptor that was created earlier.
- &cli: a pointer to a sockaddr structure where the address of the connecting client socket and the port no shall be returned.
- &cli_len: specifies the length of the supplied client sockaddr structure.

Now the server and the client can exchange data using read and write on the newfd descriptor. The write function is as shown below:

```
int nbytes = write(newfd, buff, sizeof(buff));
```

The write function takes the following arguments:

- newfd: the newly created socket descriptor in the accept phase.
- buff: a char string holding the sending message.
- sizeof(buff): the message length.

Finally, the close phase frees up the port used by the socket. The following is the code for the close function, which takes the socket descriptor as argument.

```
close(newfd);
```

Generating the Random Sequence Code

The following function will generate a random sequence code and return a character string to the connected client socket. The function takes a random count and generates a random number, which is later converted to a string:

```
string getSequence(int count) {
   string str = "";
   for (int i = 0; i < count; i += 1) {
          ostringstream ss;
```

```
            ss << rand() % 1000 + 10;
            str.append(ss.str());
    }
    return str;
}
```

Grading the Students

Because the assignment is on the cloud, there could be a huge number of students taking the course. Therefore, there should be a shell script that checks the output of the client program to see if it matches any of the lines from the server output file.

First, find all the users in the students' group by running the following code:

```
sudo apt-get install members
#for each student
for student in `members students`; do
  #get the student home directory
  loc = "\home\$student"

  #get student read file content
  student_file = 'cat $student.txt'

  #check if it matches any of the lines in
  #the server file
  for fn in 'cat server.txt'; do
    if("$fn" == "$student_file"){
      echo '$student:100' >> grades
    }
  done
done
```

The above script will first find all users in the group of students, then iterate through all students' files to see if their socket results file contains a sequence that is sent by the server by comparing it to the sever.txt file. This stores all the generated random sequence numbers that were sent to the socket clients and grades the students.

ADVANTAGES OF USING THE CLOUD

The main advantages for using the cloud to teach operating systems are the minimising of cost for providers and students. The student does not need to buy a computer meeting specific requirements; all that is needed is an average computer that can connect to the Internet to access course videos, as well as to participate in class exercises on the virtual laboratory. Also, the providers do not have to worry about increasing their laboratory computer facilities no matter how many students they have, as the cloud is scalable and can accommodate any number of students.

Another advantage for having the laboratory on the cloud is that students do not need any software, operating system or any kind of setup other than their SSH clients. One of the biggest advantage of teaching operating systems on the cloud is that it can be done using only one virtual Linux machine, which is very cost effective for the organisation conducting the laboratory sessions.

Finally, the virtual laboratory on the cloud allowed remote students to interact with other students and the teacher and to have the same experience as if they were in a traditional class.

LIMITATIONS AND SOLUTIONS

One of the drawbacks of having a virtual laboratory on the cloud is security. The instructor must ensure all the students have the right file access permissions and cannot override other student files. Another drawback is assignment grading or checking, as the instructor must create some sort of auto grading script that will check the students' assignment output without having to manually check each assignment.

CONCLUSIONS

In this article, an approach to teaching sockets, a concept of operating systems, was demonstrated to the reader. A systematic guide was provided to the reader to do the practical laboratory assignment of an operating systems class.

In summary, conducting an operating systems laboratory on the cloud allowed remote students to connect with local students and to have a similar experience to that in a physical laboratory. Also, it cut down student costs and effort, and did not require them to install Linux on their computers. Finally, the instructor was able to automate the grading process to obviate having to manually and individually grade the students.

For future course offerings, threads, processes and pipes can be added to the list of assignments. Also, there could be more sophisticated assignments requiring students to work together.

REFERENCES

1. Bojanova, I., Dimitrov, V. and Corno, F., Advancing cloud computing (guest editors' introduction). *IT Professional*, 16, **6**, 16-17 (2014), 27 May 2017, http://ieeexplore.ieee.org.adezproxy.adu.ac.ae/document/6964978/ /
2. Computer Basics: Understanding Operating Systems - Full Page. GCFLearnFree.org (2017), 27 May 2017, https://www.gcflearnfree.org/computerbasics/understanding-operating-systems/1/
3. Regehr, Why Take an Operating Systems Course?, Embedded in Academia (2017), 27 May 2017, https://blog.regehr.org/archives/164
4. Webb, K., Hibler, M., Ricci, R., Clements, A. and Lepreau, J., Implementing the emulab-planetlab portal: experience and lessons learned. *Proc. 1st Workshop Real, Large Distrib. Syst*. (2004).
5. Salah, K., Hammoud, M. and Zeadally, S., Teaching cybersecurity using the cloud. *IEEE Trans. on Learning Technologies*, 8, **4**, 383-392 (2015), 27 May 2017, http://ieeexplore.ieee.org/document/7089256/
6. Mirkovic, J. and Benzel, T., Teaching cybersecurity with deterlab. *IEEE Security Privacy*, 10, **1**, 73-76 (2012).
7. Willems, C., Klingbeil, T., Radvilavicius, L., Cenys, A. and Meinel, C., A distributed virtual laboratory architecture for cybersecurity training. *Proc. Inter. Conf. Internet Technol. Secured Trans.*, 408-415 (2011).
8. Dias, E., Application Layer and Socket Programming, LinkedIn SlideShare, 29 November (2008), 06 June 2017, https://www.slideshare.net/adorepump/application-layer-and-socket-programming-presentation