CompTIA Linux+

Customize or Write Simple Scripts

Introduction

Welcome to the **Customize or Write Simple Scripts** Practice Lab. In this module you will be provided with the instructions and devices needed to develop your hands-on skills.

Customization
Simple Scripts
Syntax

Learning Outcomes

In this module, you will complete the following exercise:

- Exercise 1 - Customize or Write Simple Scripts

After completing this lab, you will be able to:

- Use standard sh syntax
- Use command substitution
- Use the test command

Exam Objectives

The following exam objectives are covered in this lab:

- **LPI:** 103.1 Work on the command line
- **LPI:** 105.1 Customize and use the shell environment
- **CompTIA:** 5.1 Given a scenario, deploy and execute basic BASH scripts.

Note: *Our main focus is to cover the practical, hands-on aspects of the exam objectives. We recommend referring to course material or a search engine to research theoretical topics in more detail.*

Lab Duration

It will take approximately **1 hour** to complete this lab.

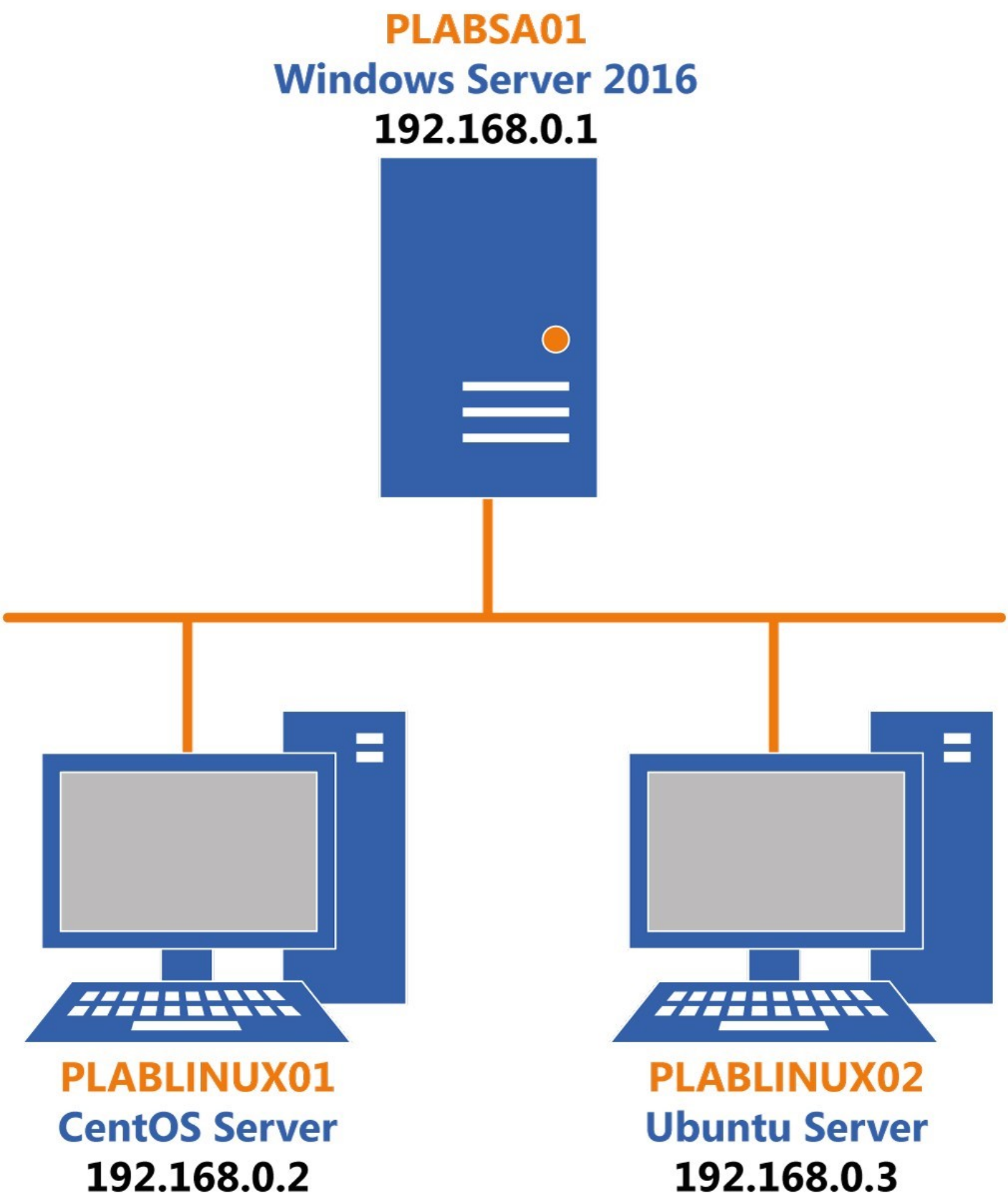
Help and Support

For more information on using Practice Labs, please see our **Help and Support** page. You can also raise a technical support ticket from this page.

Click Next to view the Lab topology used in this module.

Lab Topology

During your session, you will have access to the following lab configuration.



Depending on the exercises you may or may not use all of the devices, but they are shown here in the layout to get an overall understanding of the topology of the lab.

- **PLABSA01** (Windows Server 2016)
- **PLABLINUX01** (CentOS Server)
- **PLABLINUX02** (Ubuntu Server)

Click Next to proceed to the first exercise.

Exercise 1 - Customize or Write Simple Scripts

Scripts allow you to automate a number of tasks or allow multiple commands to be executed at once. Assume that you have to add a few thousands of users. One method is to execute the useradd command thousands of times or simply write a script that will create these thousands of users without any user intervention. Scripts are especially relevant while executing loops and tests, that too without user intervention. In this exercise, you will understand how to write or customize simple scripts.

Please refer to your course material or use your preferred search engine to research this topic in more detail.

Learning Outcomes

After completing this exercise, you will be able to:

- Log into a Linux System
- Use standard sh syntax
- Use command substitution
- Use the test command

Your Devices

You will be using the following device in this lab. Please power these on now.

- **PLABLINUX01** (CentOS Server)

Task 1 - Use standard sh syntax

The script files are stored with an extension of **.sh**. Each script contains a script interpreter that appears as a commented line in the script. In this task, you will create and execute simple script and scripts to implement loops. You will also learn how to execute the loops without writing shell scripts.

To use the standard sh syntax, perform the following steps:

Step 1

On the desktop, right-click and select **Open Terminal**.

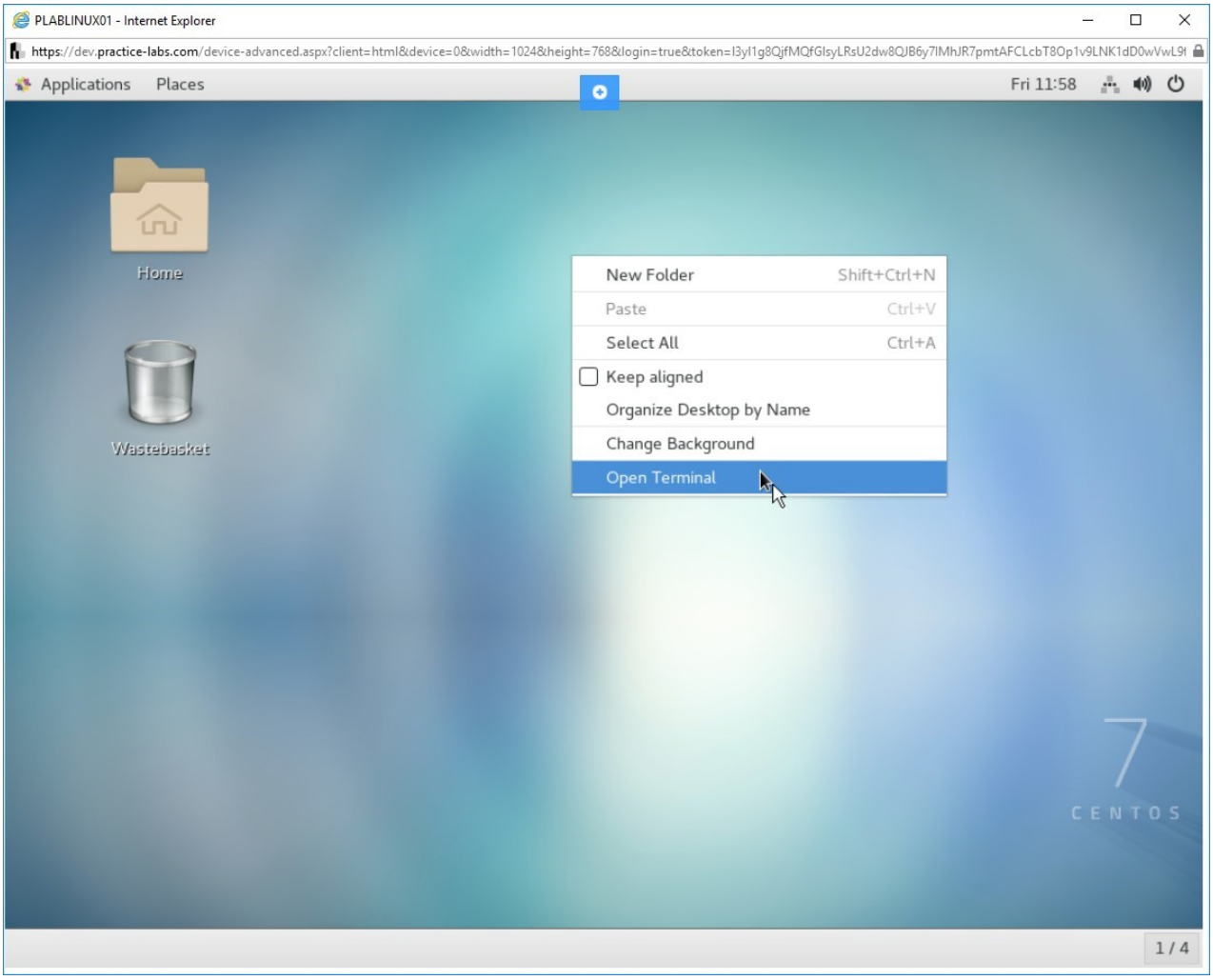


Figure 1.1 Screenshot of PLABLINUX01: Selecting the Open Terminal option from the context menu.

Step 2

The command prompt window is displayed. Type the following command:

```
su -
```

Press **Enter**.

At the **Password** prompt, type the following password:

```
Passw0rd
```

Press **Enter**.

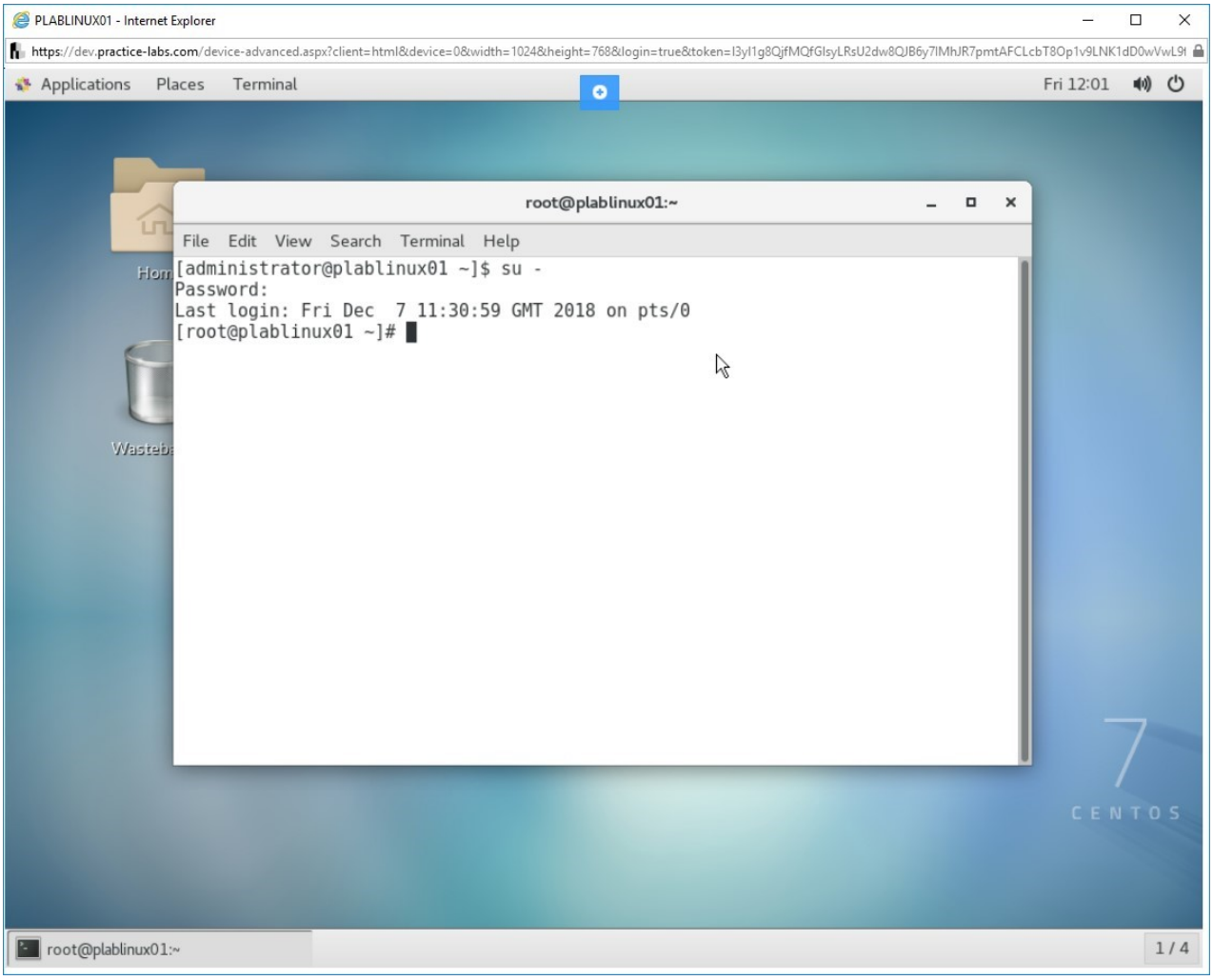


Figure 1.2 Screenshot of PLABLINUX01: Changing the account to the root account with the su command.

Step 3

Clear the screen by entering the following command:

```
clear
```

> ***Note**: The clear command is used before every step to enable the learners to get a clear view of the output of each command. Otherwise, it is not mandatory to use the clear command before every command.*

You can create a script using the vi editor.

To create a script, type the following command:

```
vi plab.sh
```

Press **Enter**.

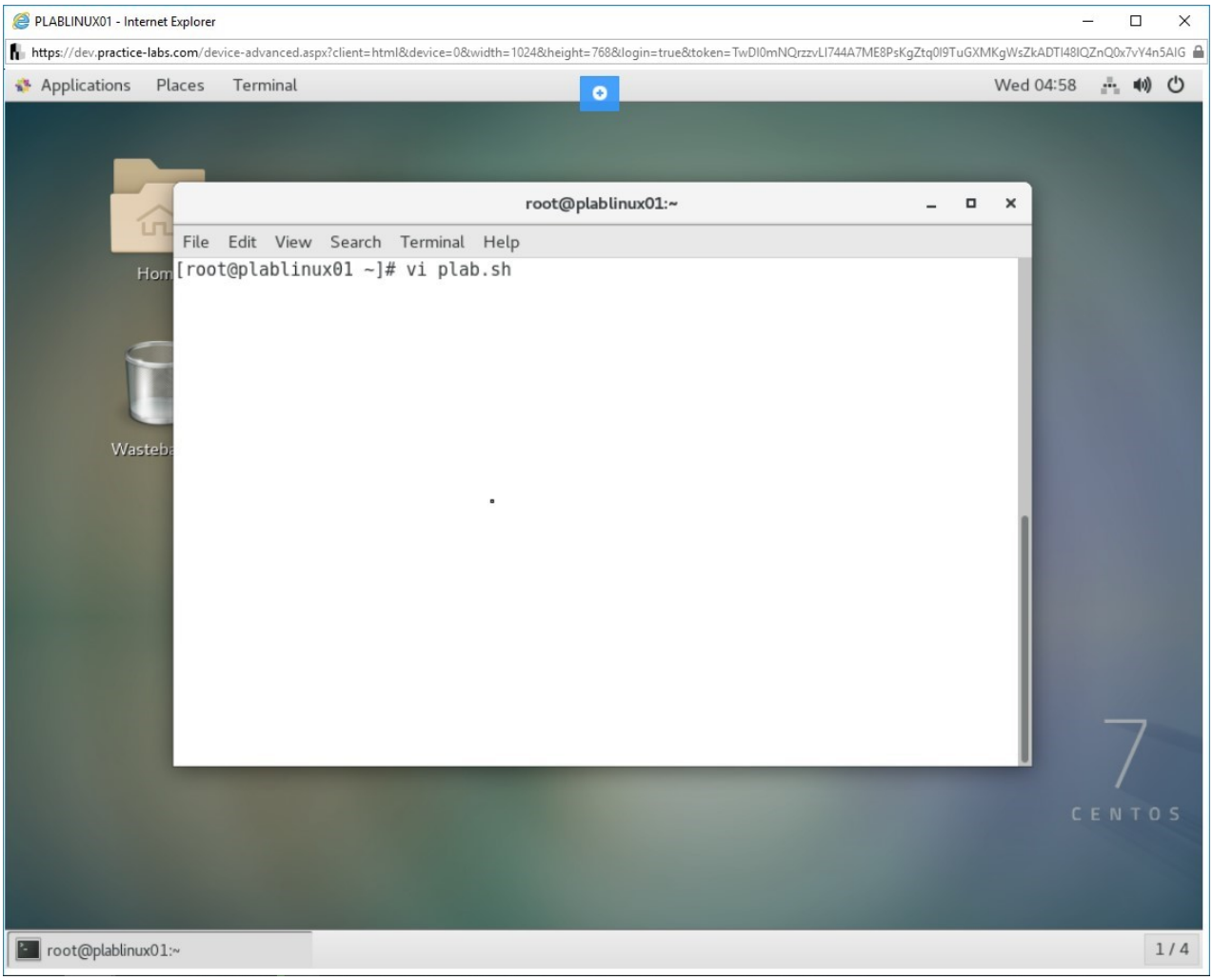


Figure 1.3 Screenshot of PLABLINUX01: Creating a script using the vi command.

The vi editor is displayed.

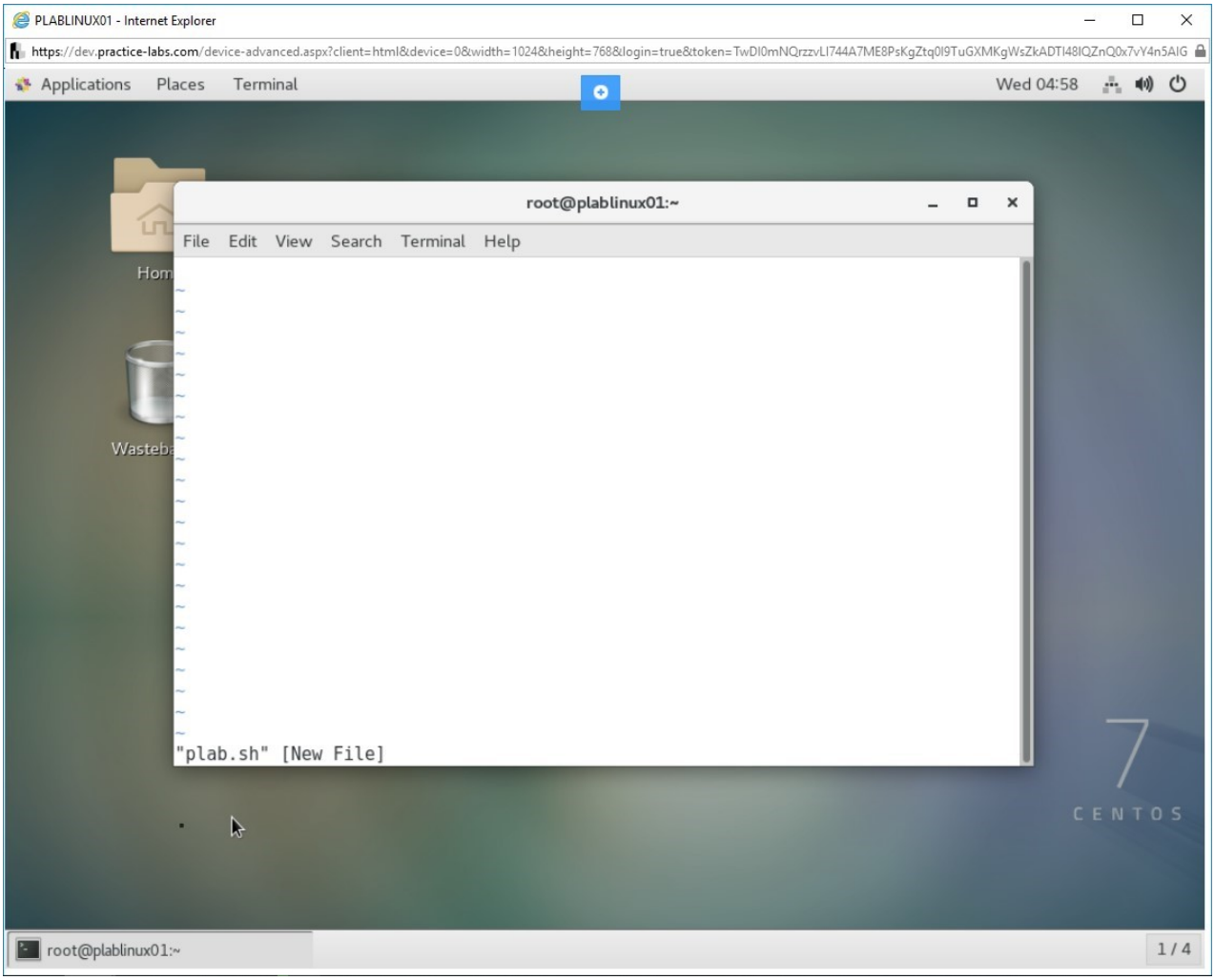


Figure 1.4 Screenshot of PLABLINUX01: Displaying the vi editor window.

Step 4

Press **i** to invoke the insert mode.

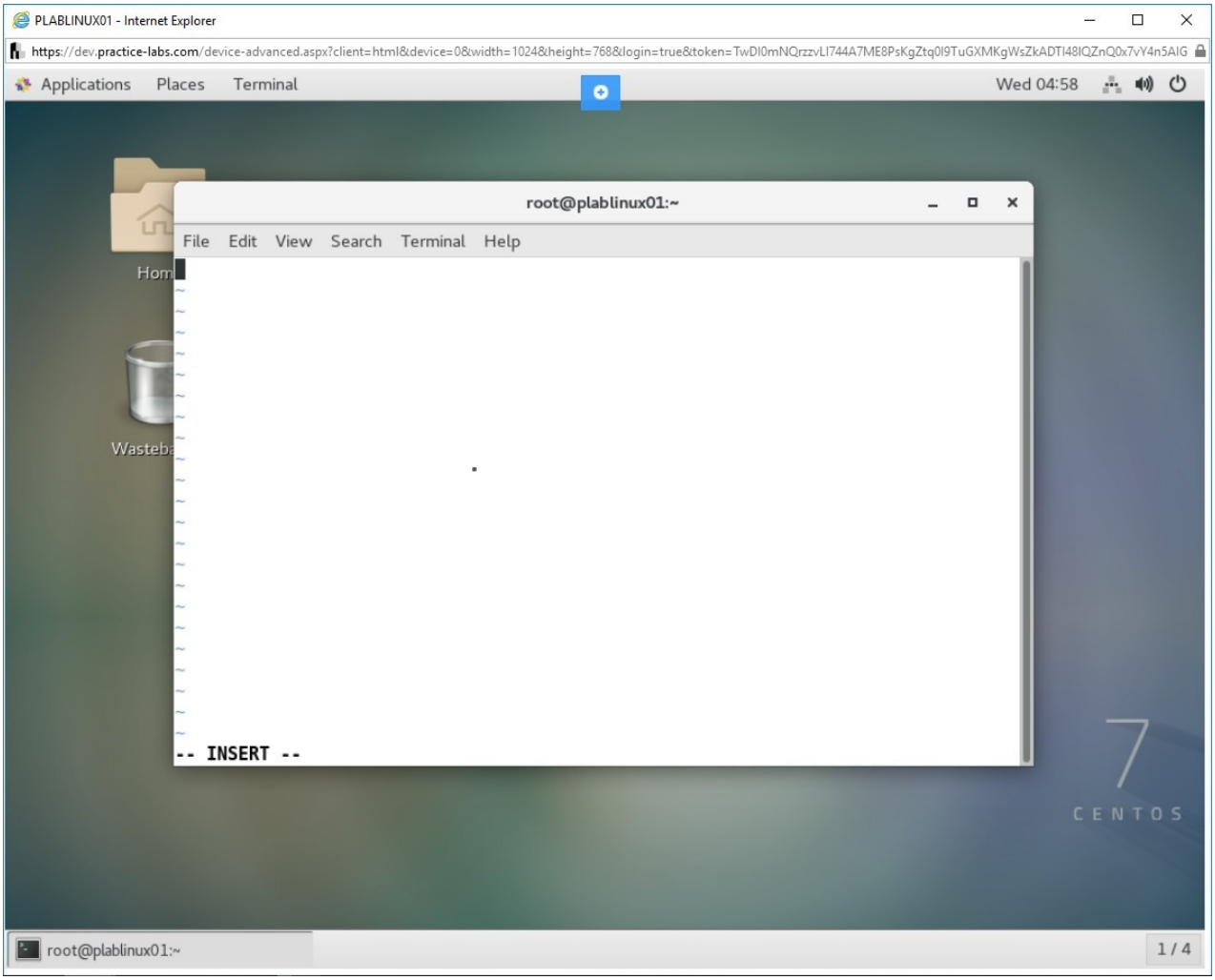


Figure 1.5 Screenshot of PLABLINUX01: Entering the insert mode in the vi editor.

Step 5

While creating a script, you need to first define the script interpreter. The commonly used interpreters include **bash** or **sh**, although other interpreters are available.

For this task, you specify bash as the script interpreter. Type the following command:

```
#!/bin/bash
```

Press **Enter**.

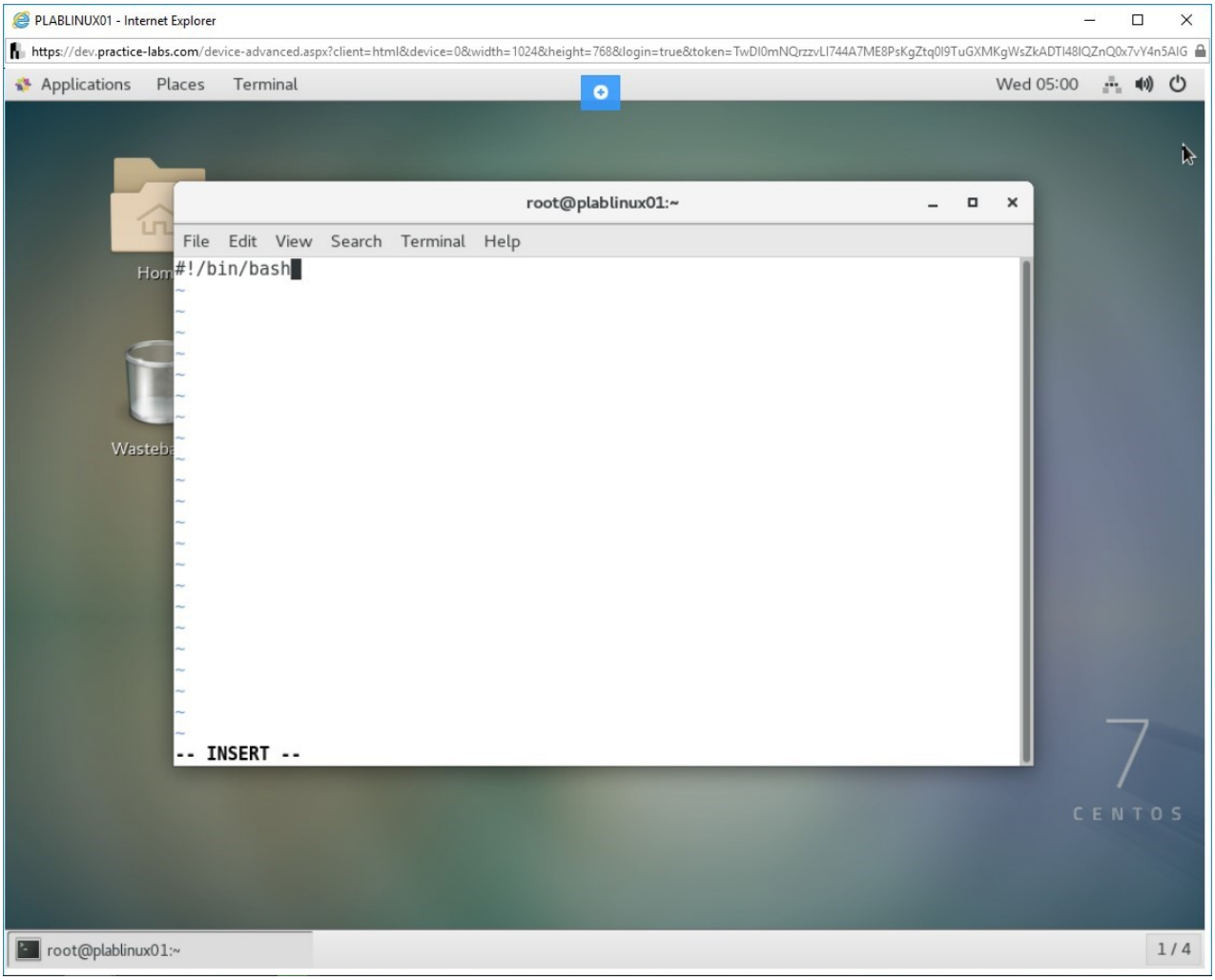


Figure 1.6 Screenshot of PLABLINUX01: Defining the script interpreter.

Step 6

In the next line, type the following command:

```
echo "hello world"
```

Press **Enter**.

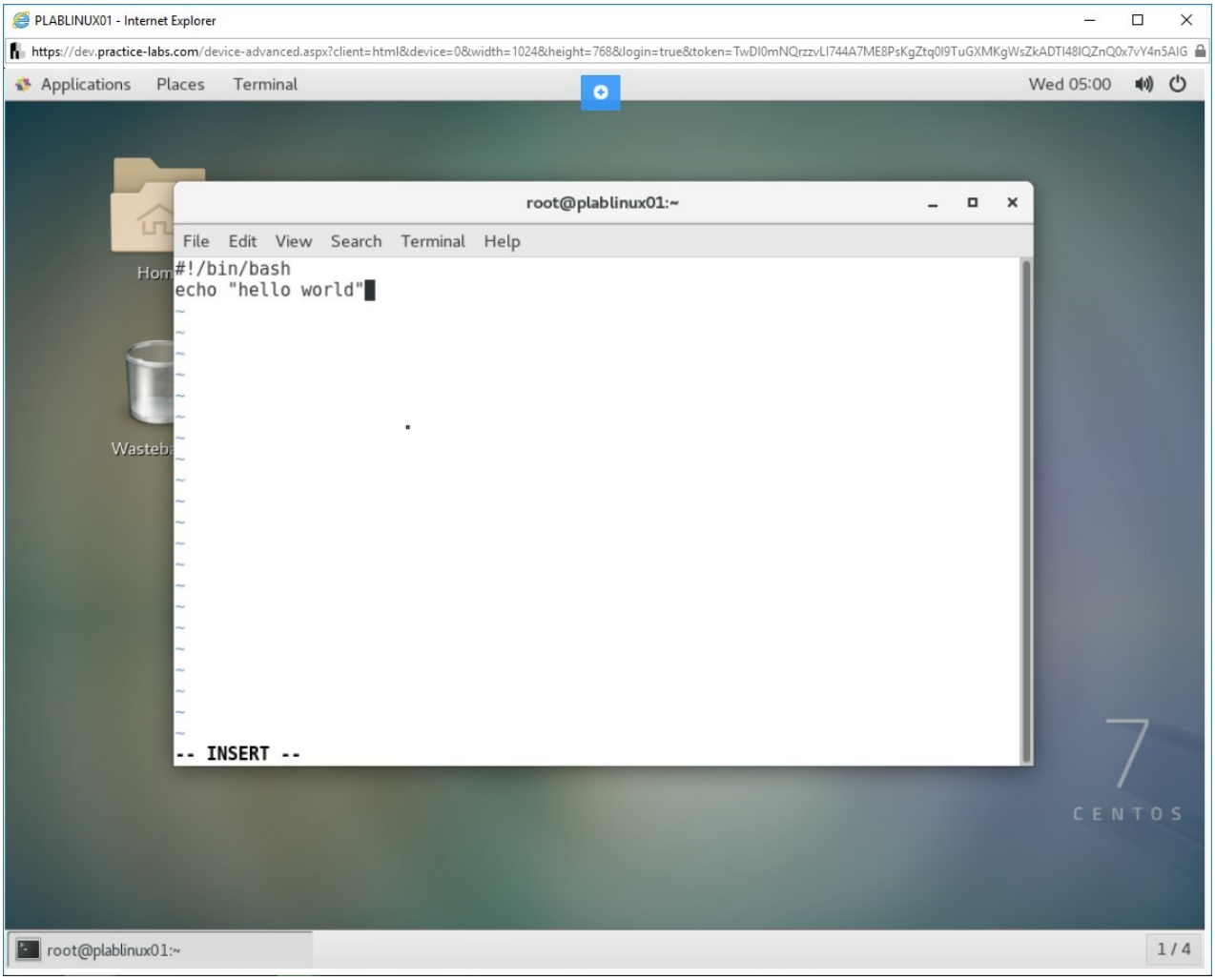


Figure 1.7 Screenshot of PLABLINUX01: Entering the script statements.

Step 7

Now, you need to save the file.

To save the file, press **ESC** and then type the following command:

```
:wq
```

Press **Enter**.

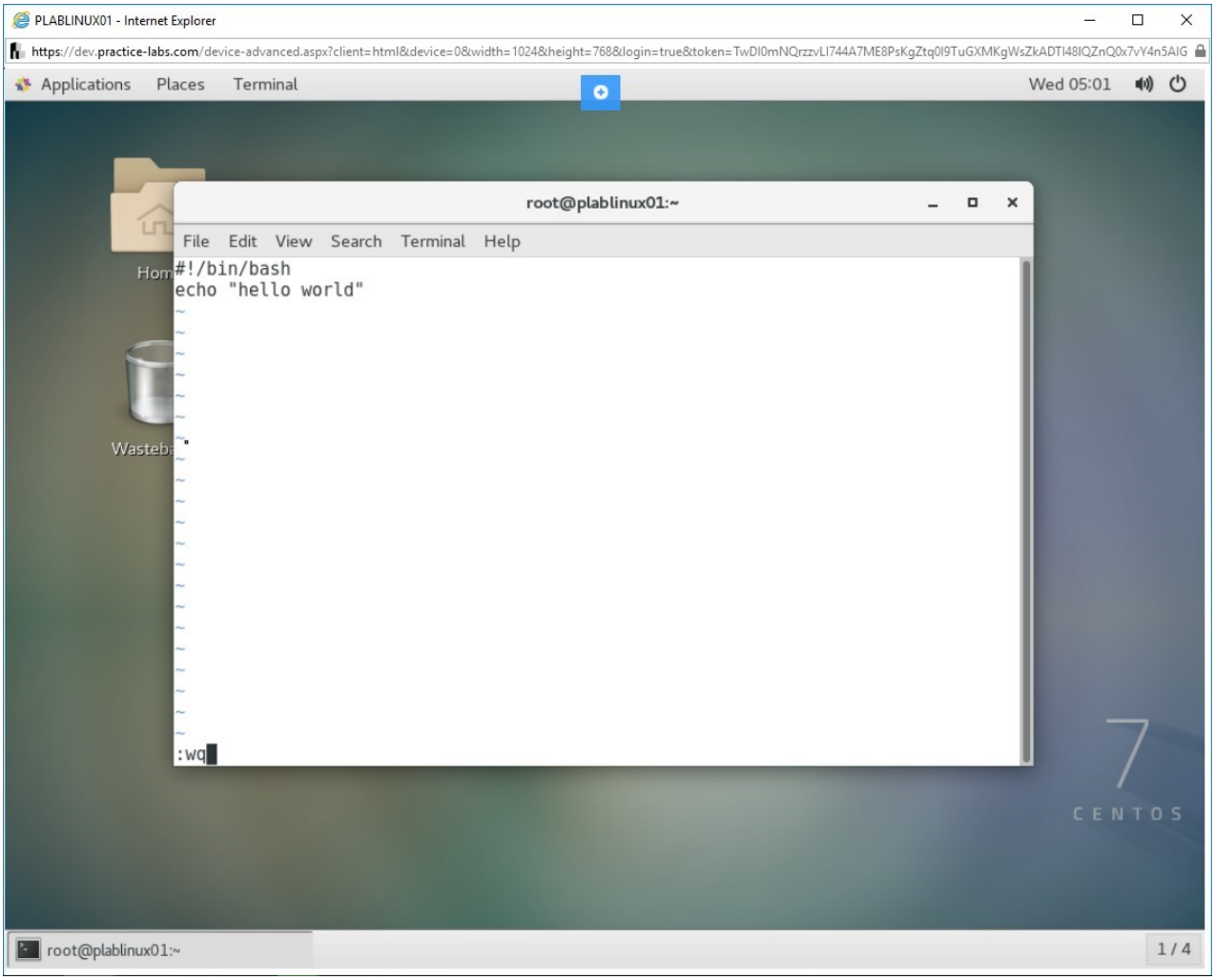


Figure 1.8 Screenshot of PLABLINUX01: Saving and closing the file.

Step 8

You are back on the command prompt.

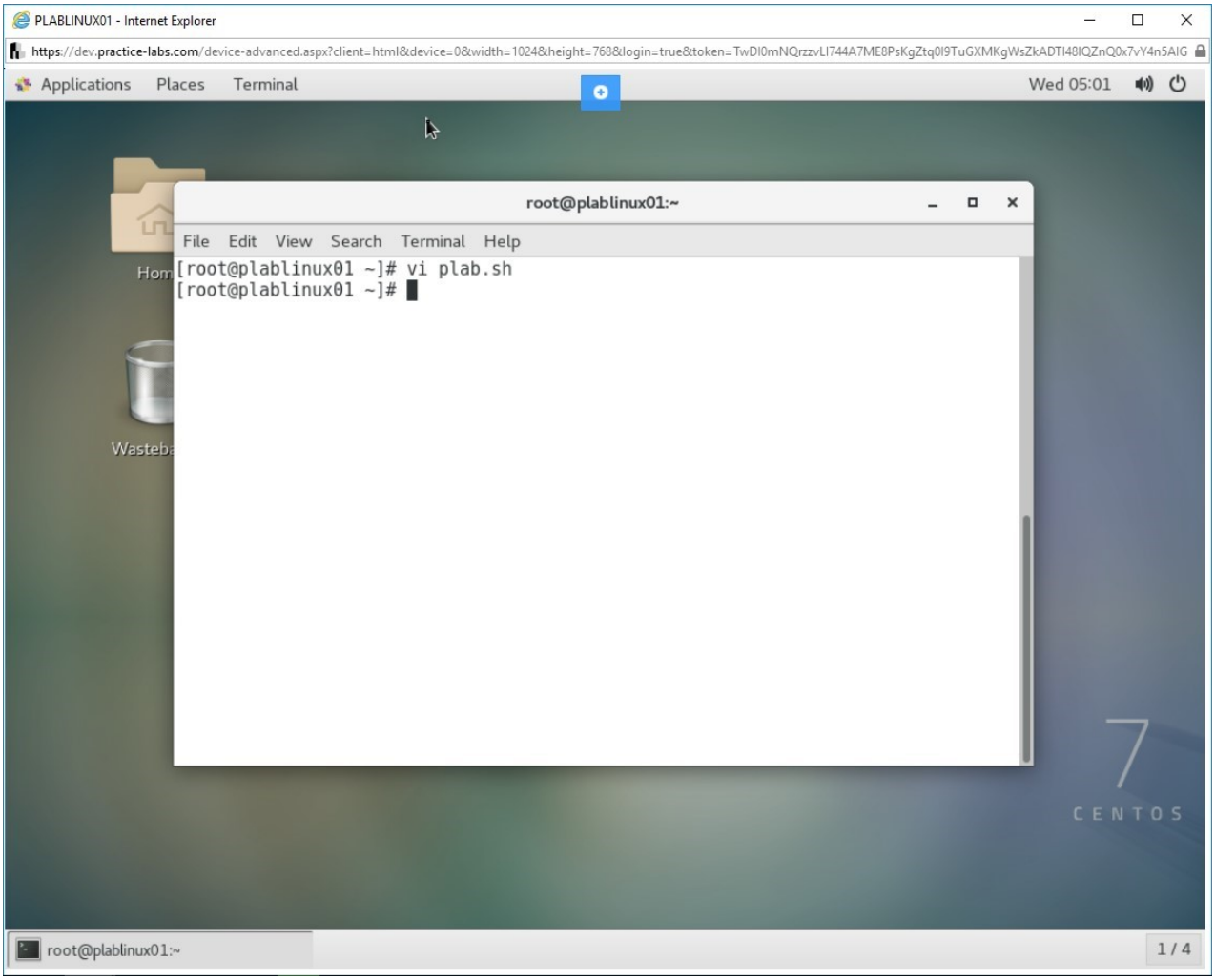


Figure 1.9 Screenshot of PLABLINUX01: Displaying the command prompt.

Step 9

To verify that the script is created, enter the command

```
ls
```

Notice that the **plab.sh** file is listed in the **root** directory.

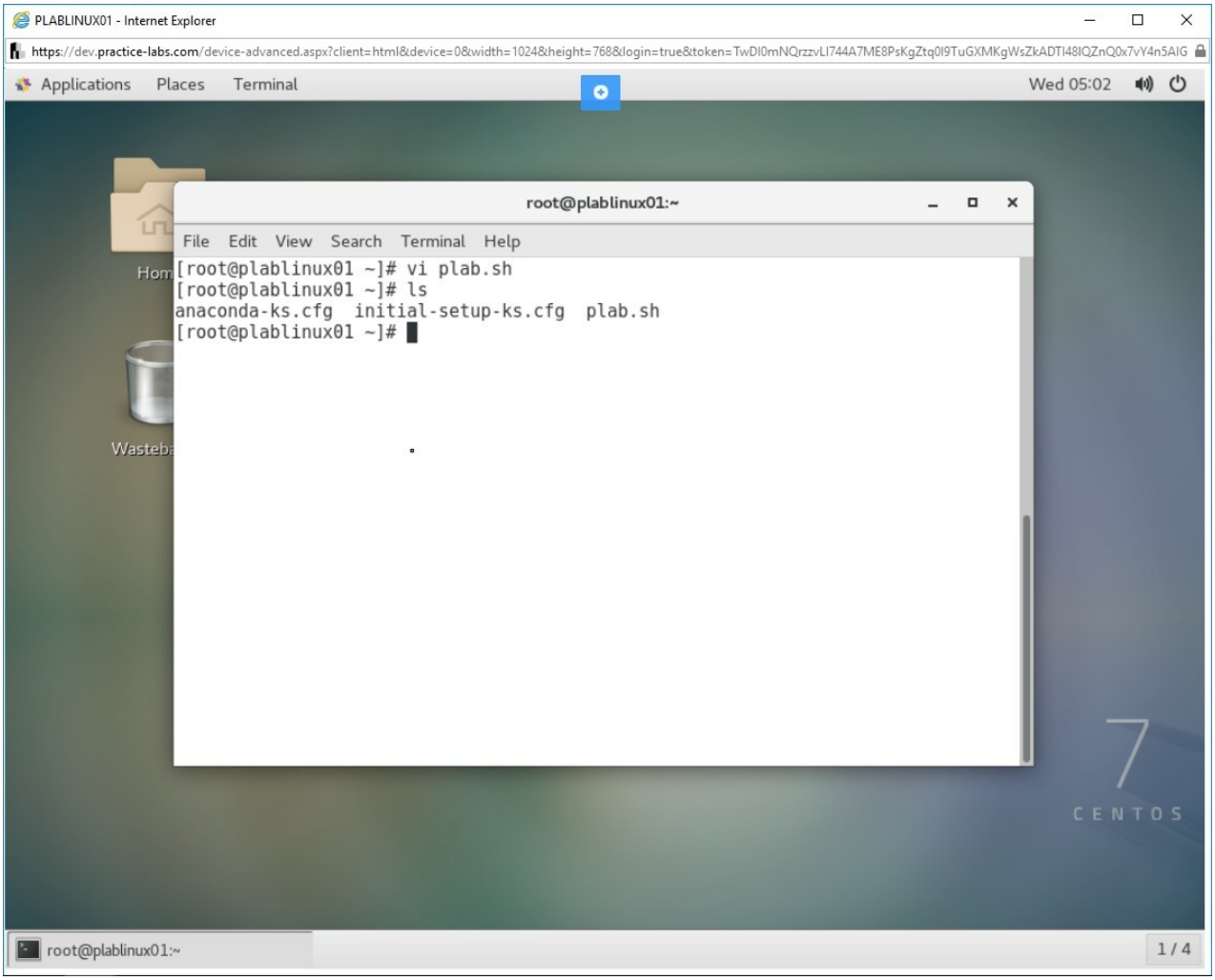


Figure 1.10 Screenshot of PLABLINUX01: Listing the files in the root directory.

Step 10

Clear the screen by entering the following command:

```
clear
```

To run the bash scripts, type the following command:

```
bash plab.sh
```

Press **Enter**.

Note that the script runs successfully.

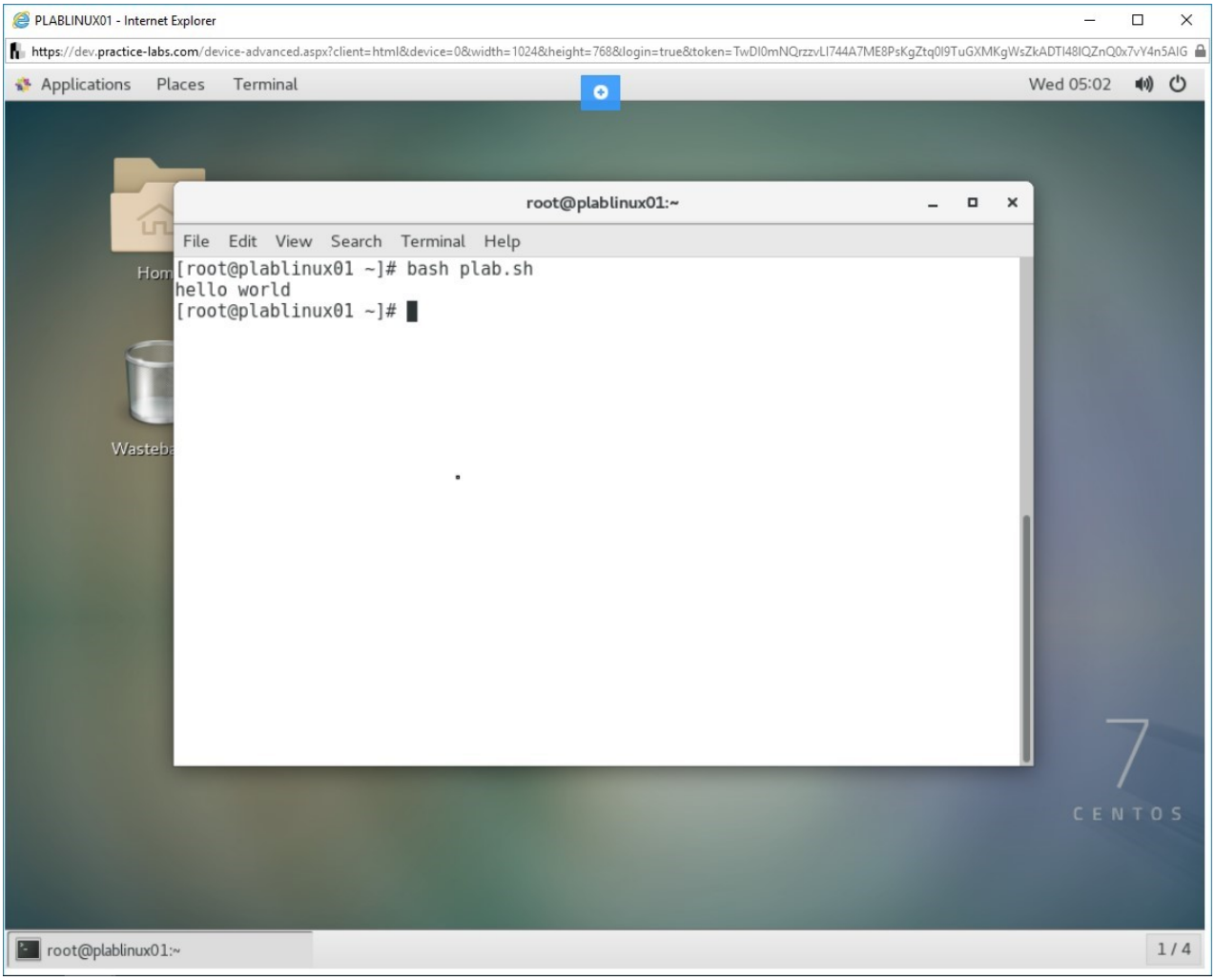


Figure 1.11 Screenshot of PLABLINUX01: Executing the script using the bash command.

Step 11

The scripts with executable permissions, can be run by prefixing the **./** with their names. For example, to execute **plab.sh** without the help of bash, you can enter **./plab.sh**

To run **plab.sh**, type the following command:

```
./plab.sh
```

Press **Enter**.

Note that there is an error.

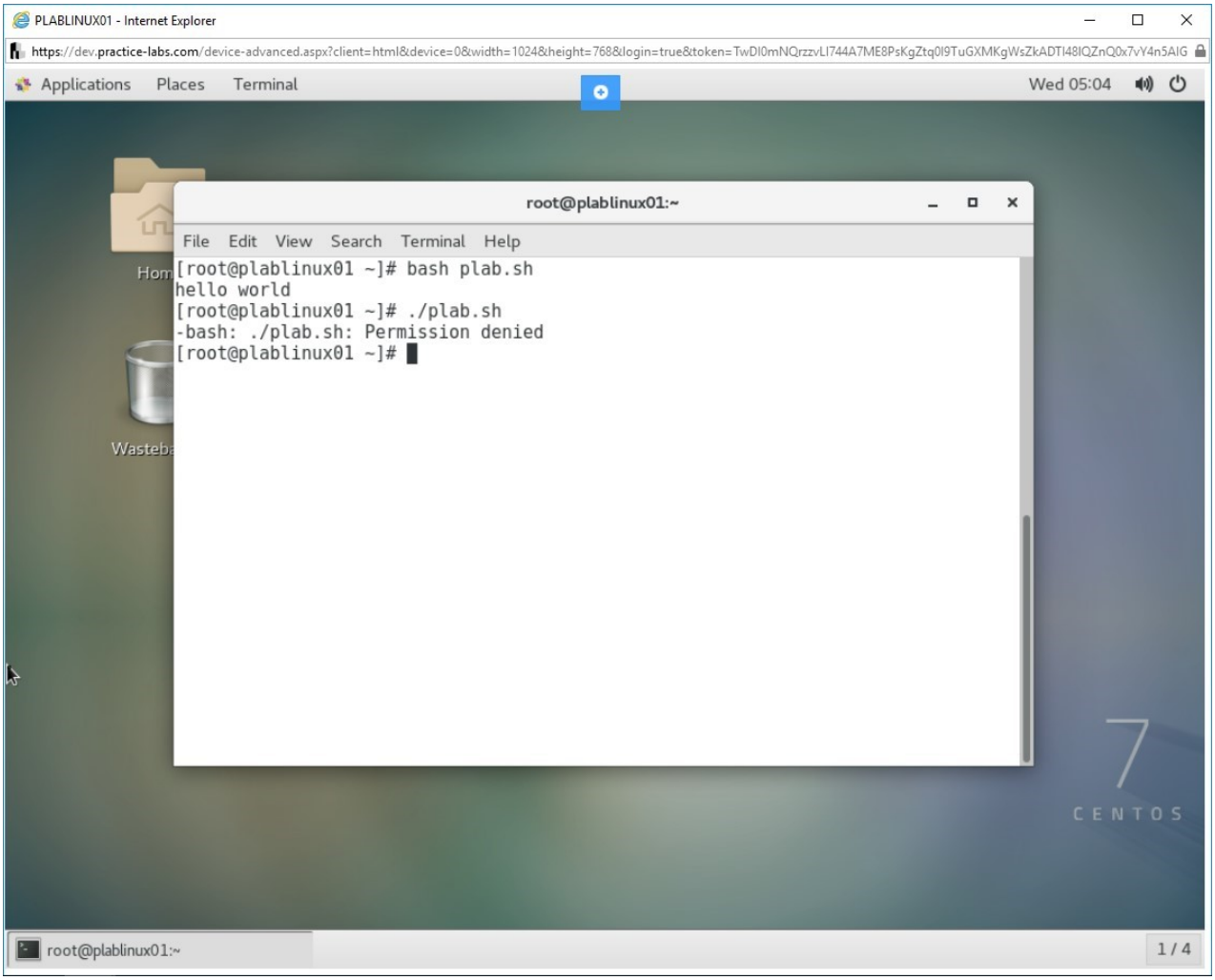


Figure 1.12 Screenshot of PLABLINUX01: Displaying the error after executing the script.

Step 12

Let's now change the permissions on the scripts to allow everyone to execute this script. To add permissions, type the following command:

```
chmod +x plab.sh
```

Press **Enter**.

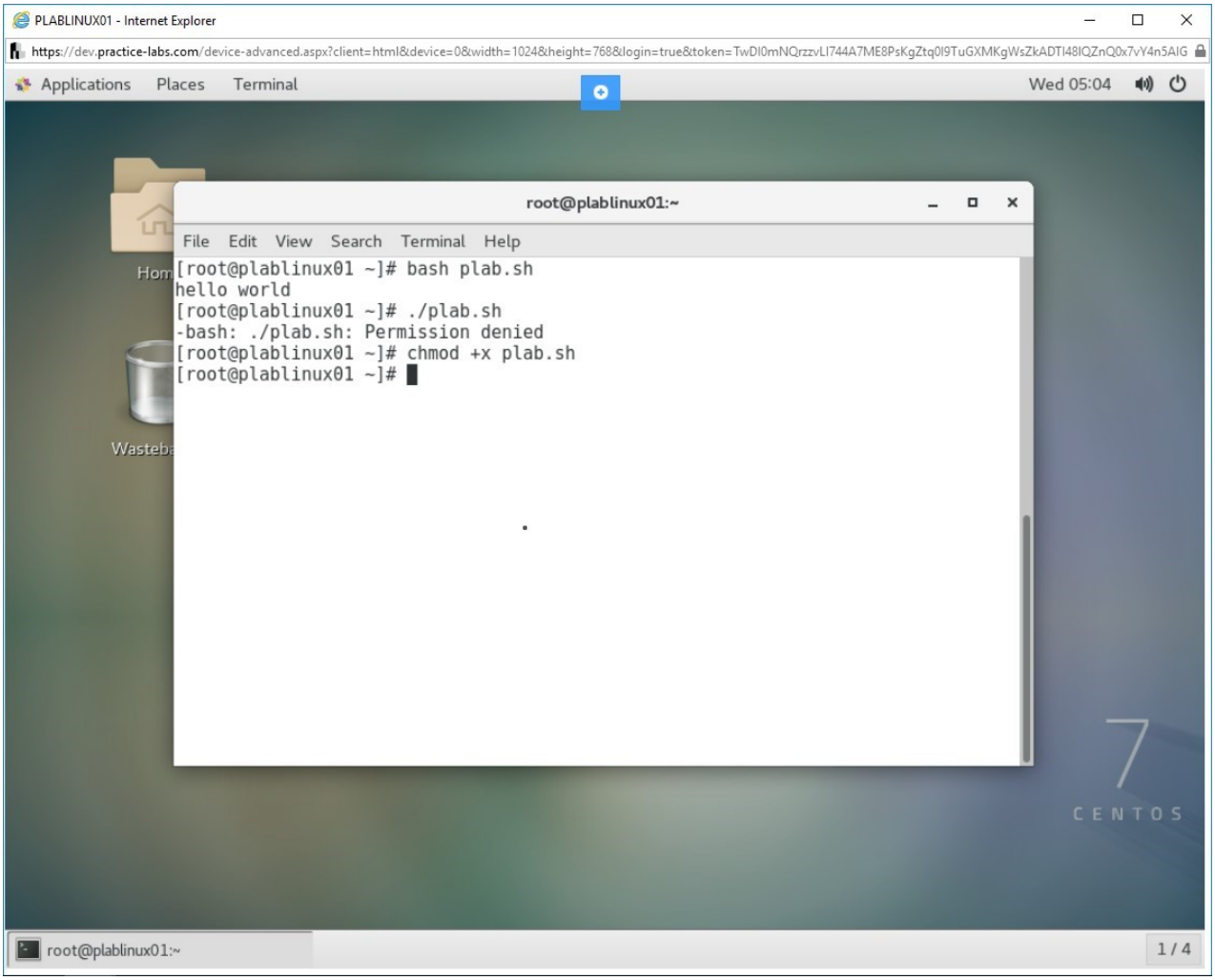


Figure 1.13 Screenshot of PLABLINUX01: Changing the permission on the script.

Step 13

Clear the screen by entering the following command:

```
clear
```

To run the **plab.sh** script, type the following command:

```
./plab.sh
```

Press **Enter**.

Notice that the script is running now.

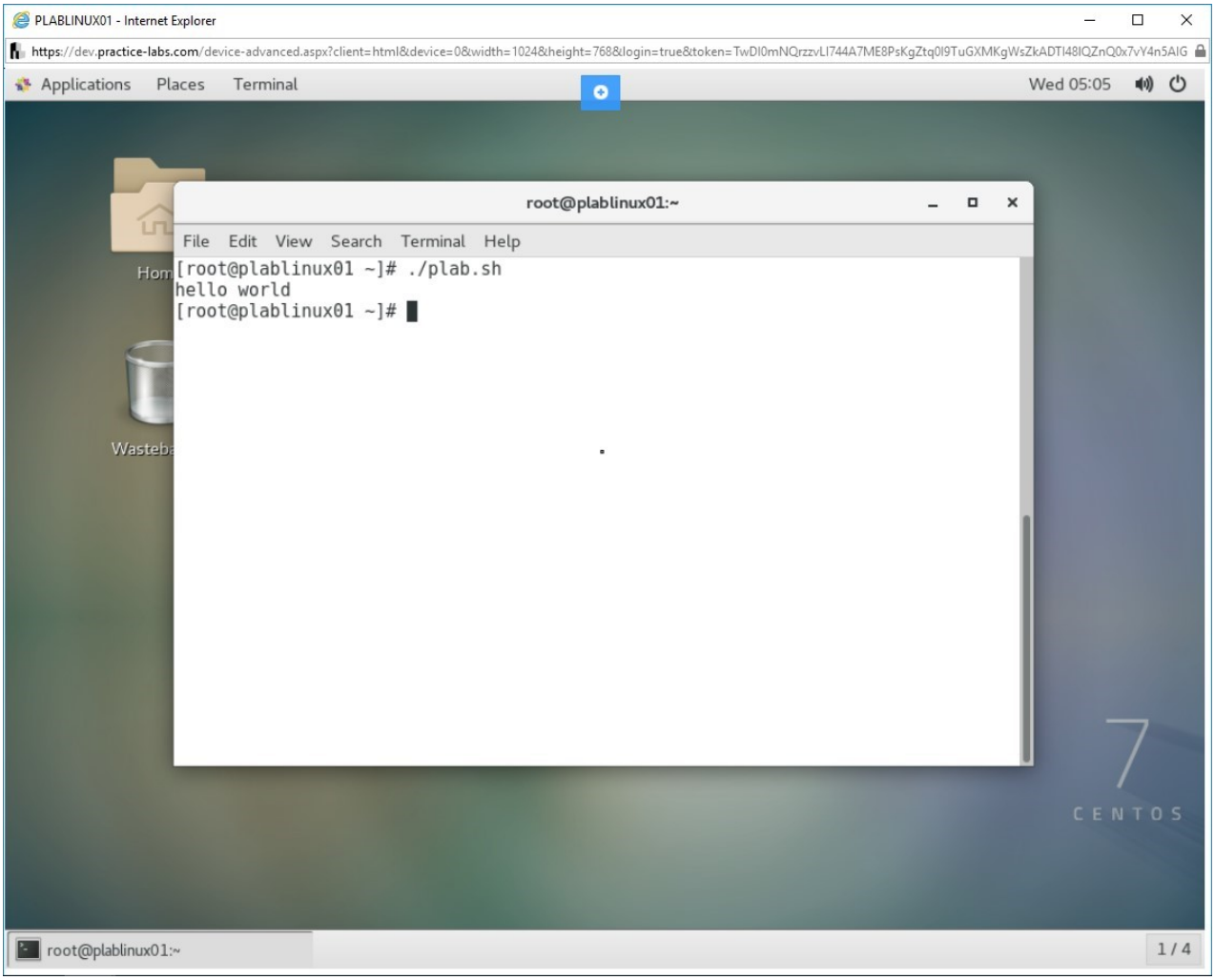


Figure 1.14 Screenshot of PLABLINUX01: Running the script.

Step 14

Clear the screen by entering the following command:

```
clear
```

You can also create a script to execute a loop. For example, you can write a script to create 10 users without having to enter the **useradd** command 10 times. You will use the **for** loop to do this.

To use the vi editor to create a new script with the name **plabuser.sh**, type the following command:

```
vi plabuser.sh
```

Press **Enter**.

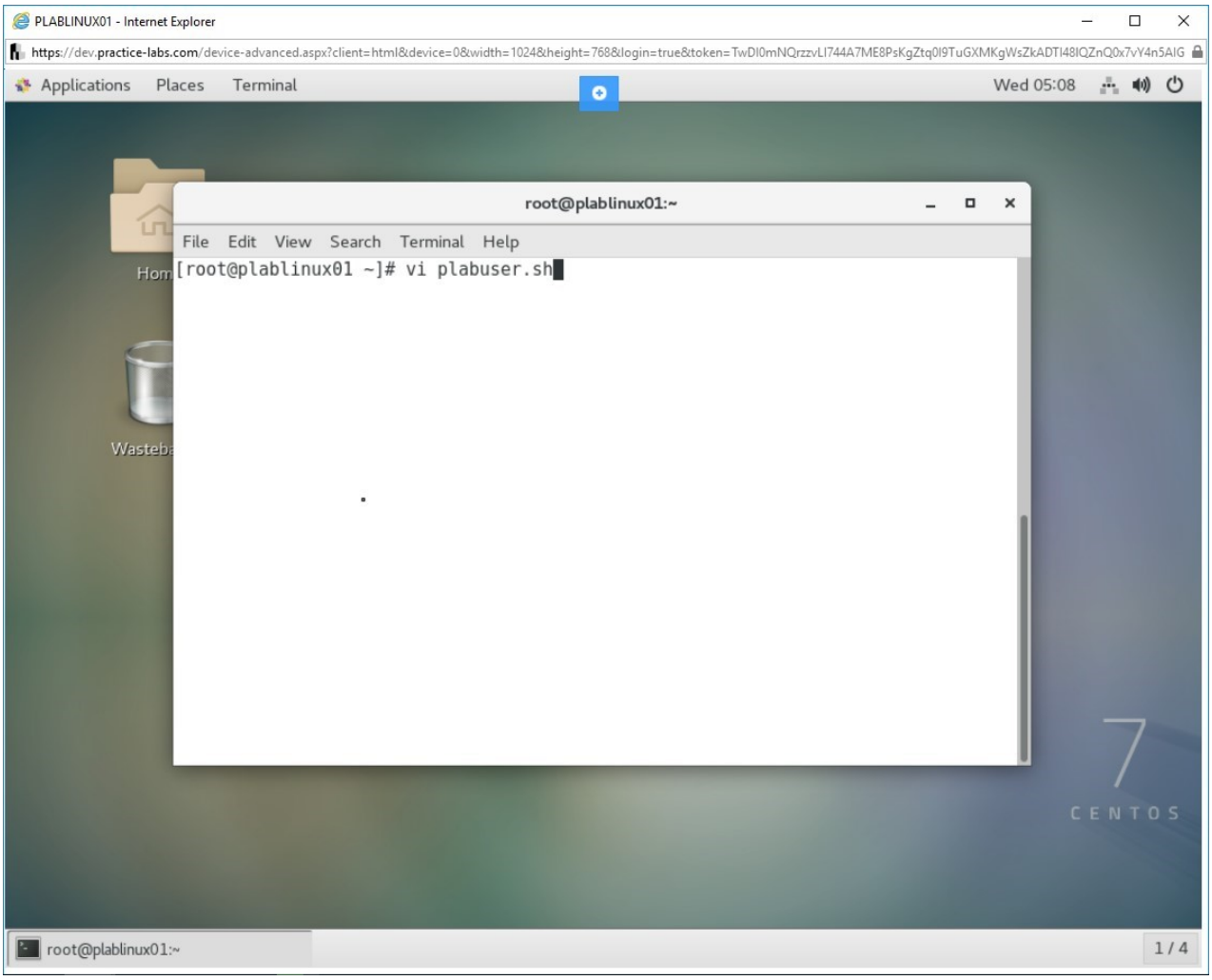


Figure 1.15 Screenshot of PLABLINUX01: Creating a script using the vi editor.

Step 15

The **vi** editor is displayed.

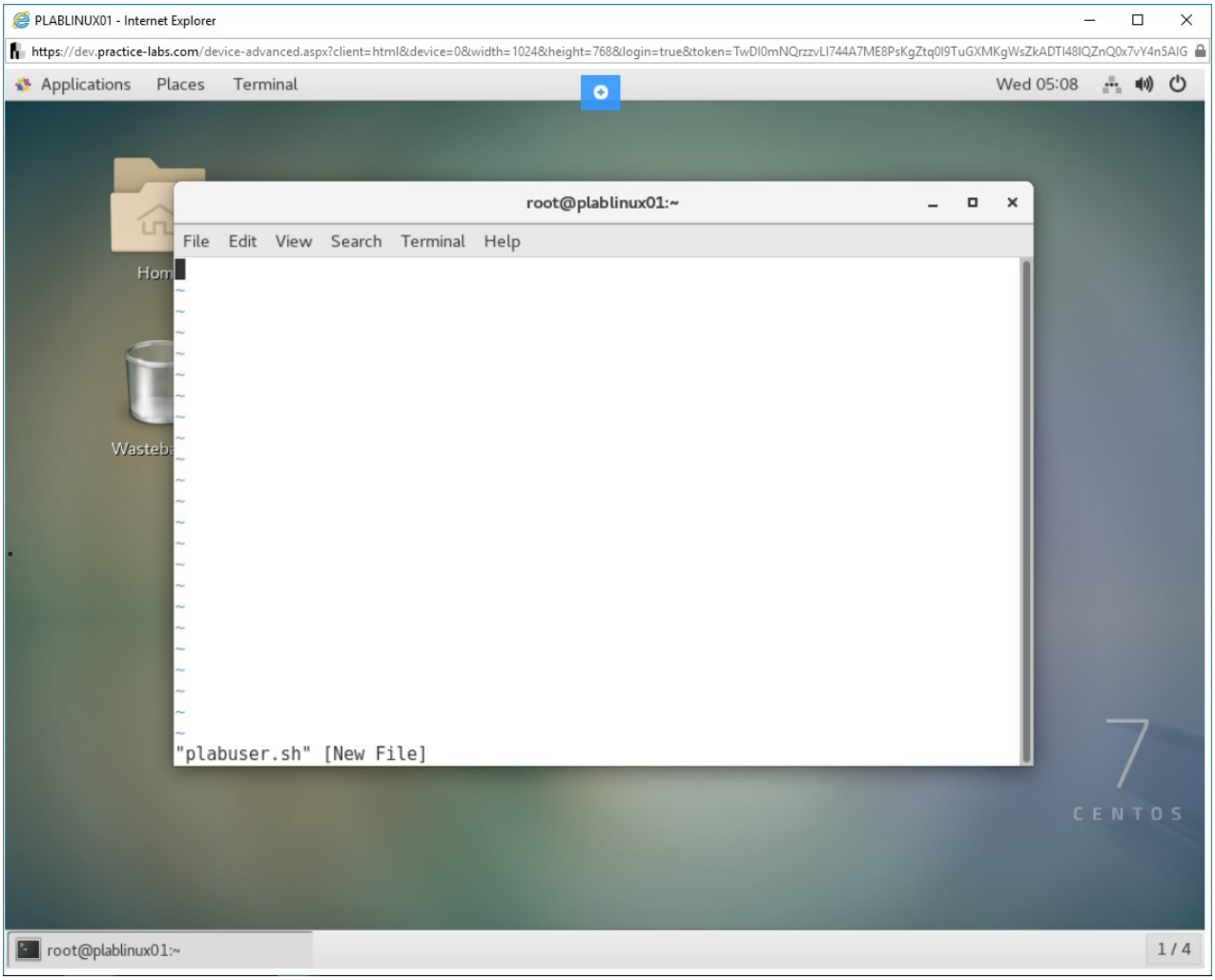


Figure 1.16 Screenshot of PLABLINUX01: Displaying the vi editor.

Step 16

Press **i** to invoke the insert mode.

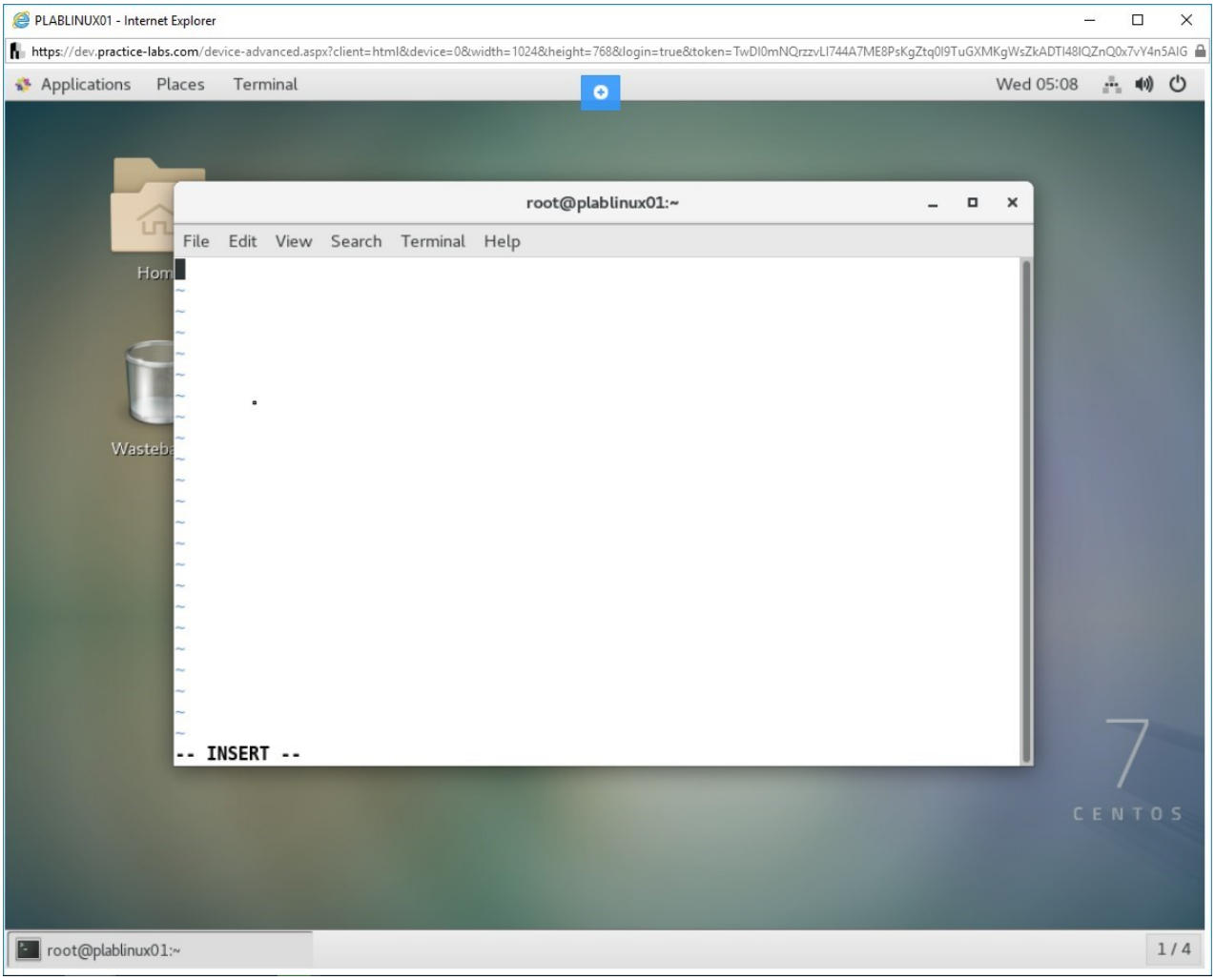


Figure 1.17 Screenshot of PLABLINUX01: Entering the insert mode.

Step 17

To specify the script interpreter, type the following command:

```
#!/bin/bash
```

Press **Enter**.

After that, type the following lines:

```
for u in $(seq 10) ; do
useradd user$u
echo user${u}
done
```

This script creates **10** users with the names starting with **user1**, **user1**, and so on.

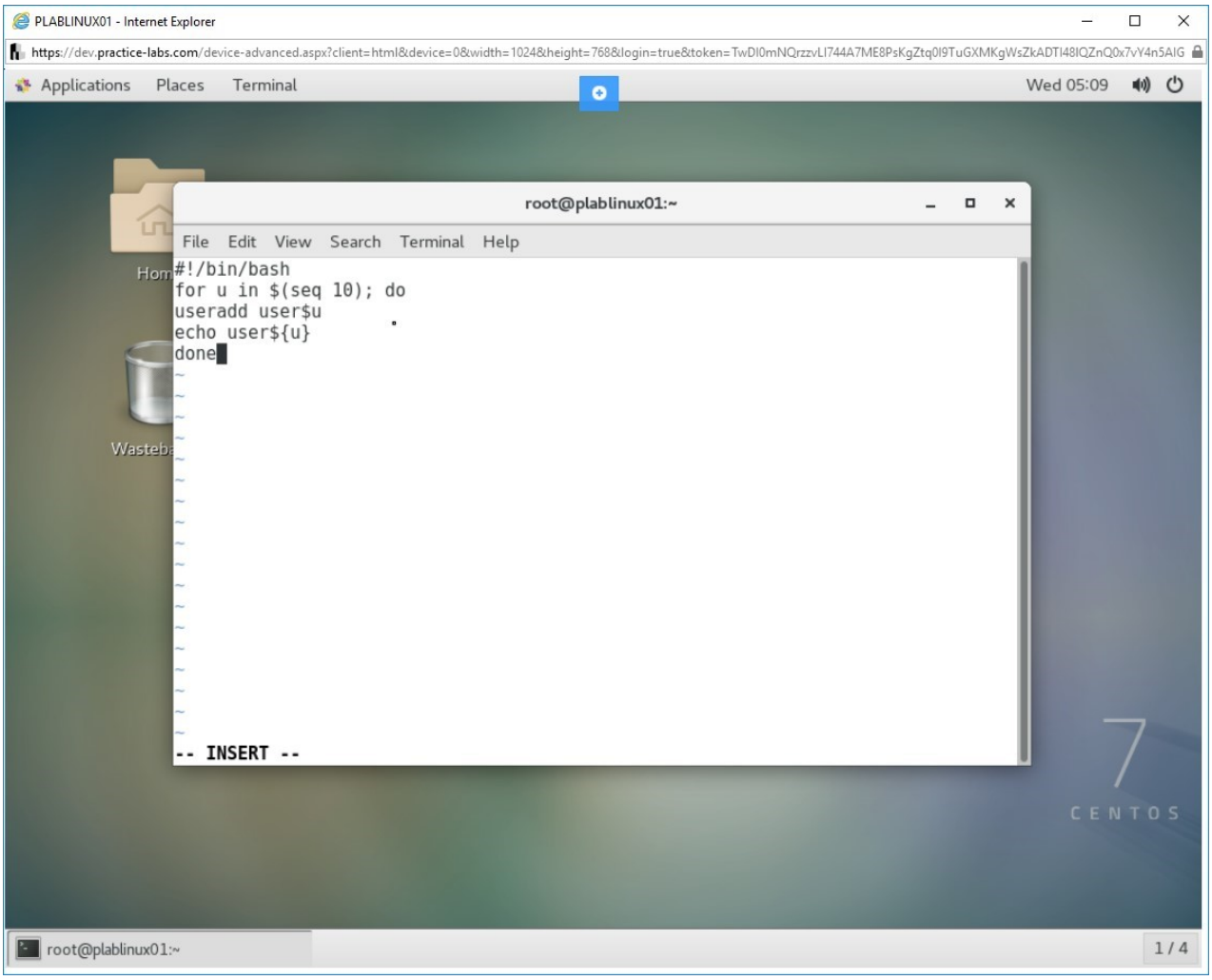


Figure 1.18 Screenshot of PLABLINUX01: Entering the statements in the script.

Step 18

Now, you need to save the file. To save the file, press **ESC** and then type the following command:

```
:wq
```

Press **Enter**.

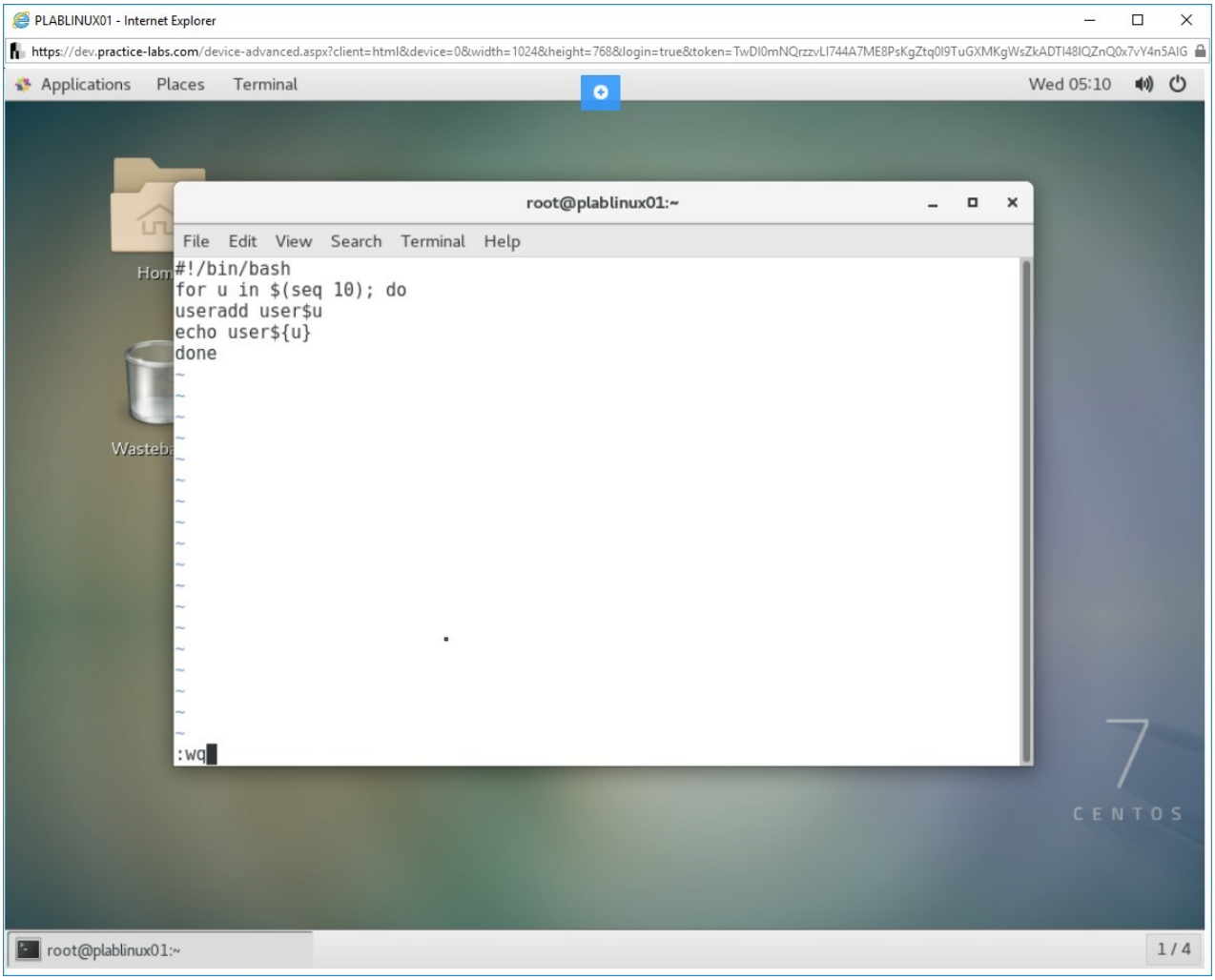


Figure 1.19 Screenshot of PLABLINUX01: Saving and exiting the file.

Step 19

You are back on the command prompt.

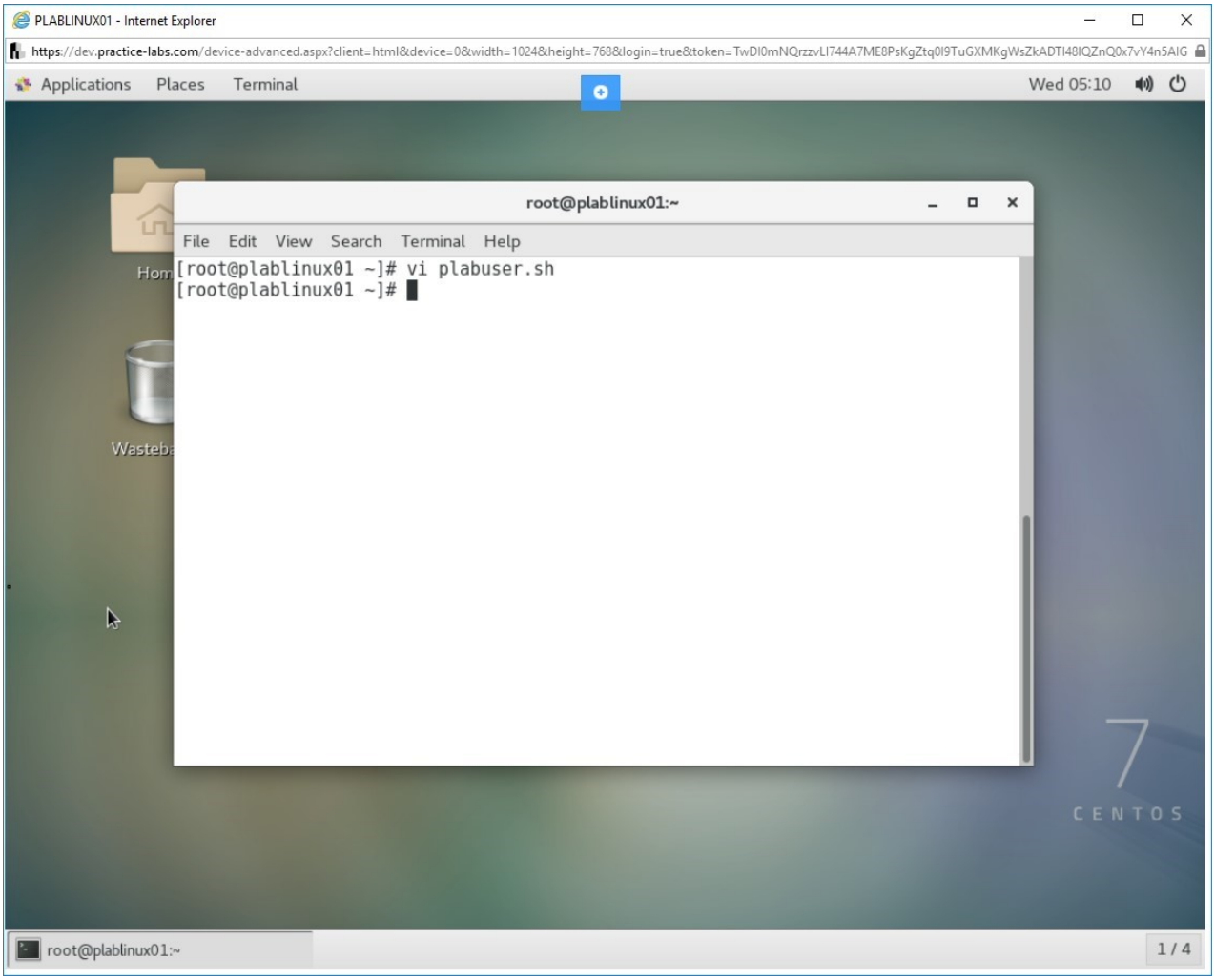


Figure 1.20 Screenshot of PLABLINUX01: Displaying the command prompt.

Step 20

Clear the screen by entering the following command:

```
clear
```

To execute the **plabuser.sh** script, you change the permissions given to the script**.**

To change permissions, type the following command:

```
chmod +x plabuser.sh
```

Press **Enter**.

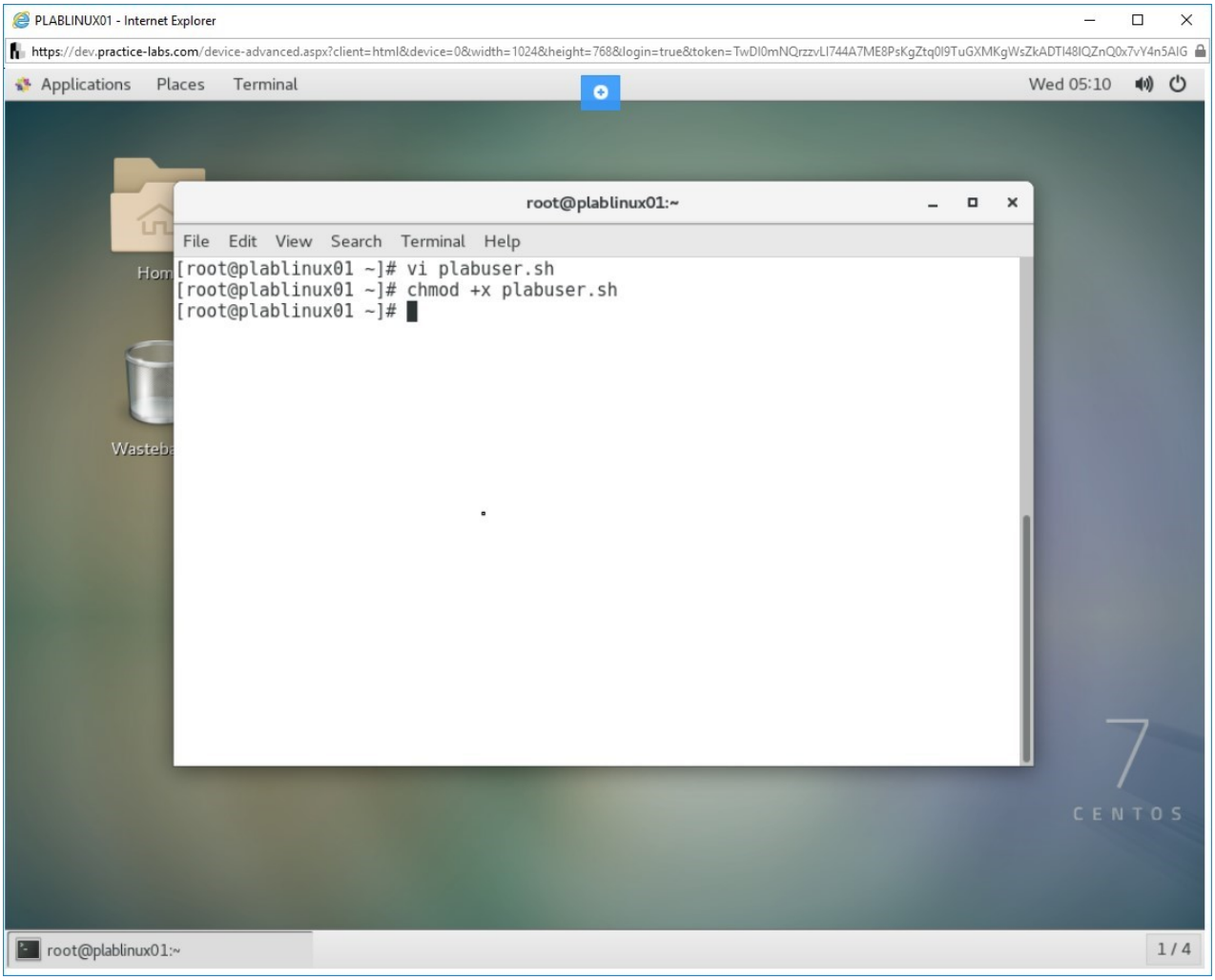


Figure 1.21 Screenshot of PLABLINUX01: Changing the script permissions.

Step 21

Now, you run the script using the following command:

```
./plabuser.sh
```

Note that **10** users have been created.

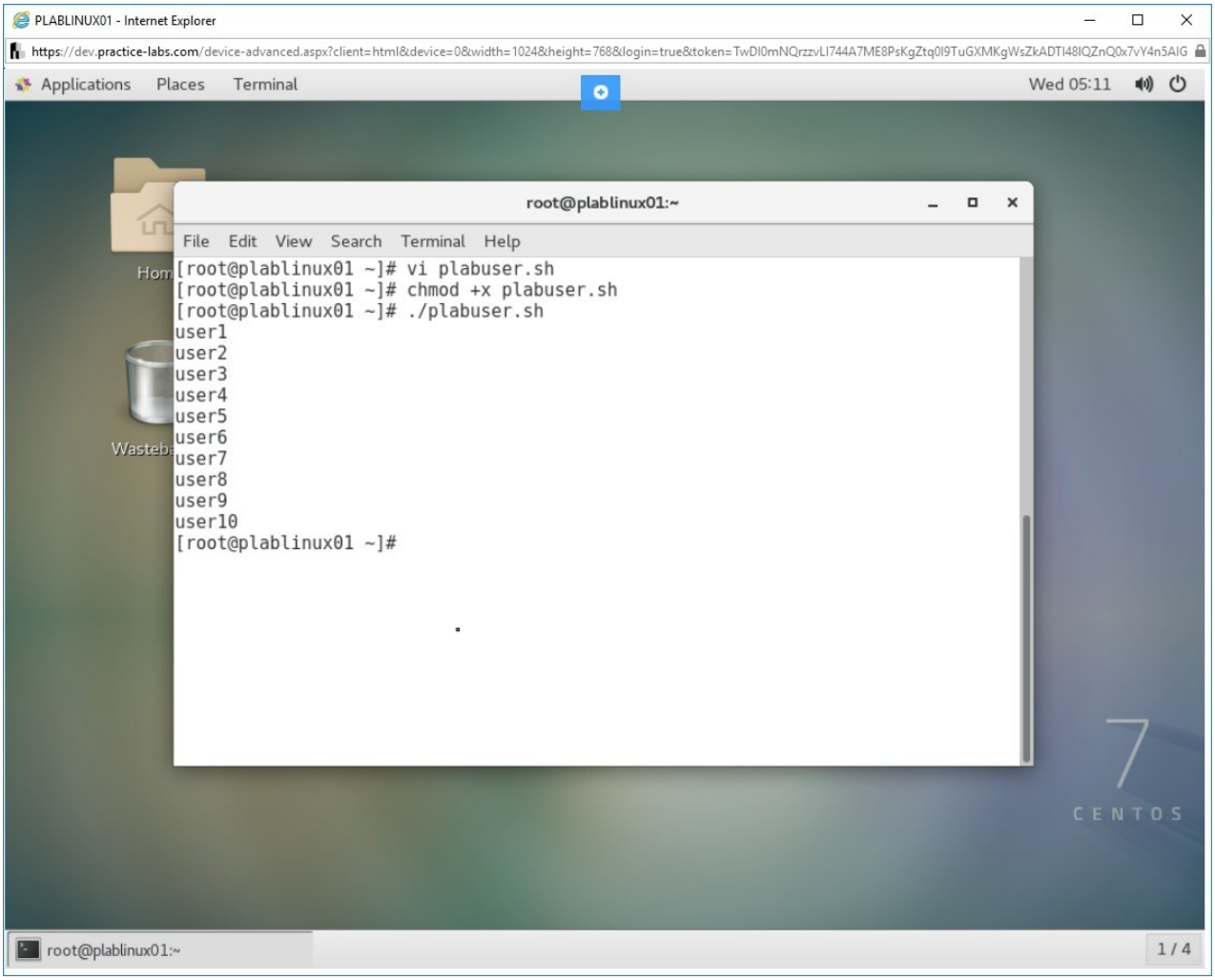


Figure 1.22 Screenshot of PLABLINUX01: Executing the script.

Step 22

Clear the screen by entering the following command:

```
clear
```

To verify that the users and their home directories have been created, type the following command:

```
ls -l /home
```

Press **Enter**.

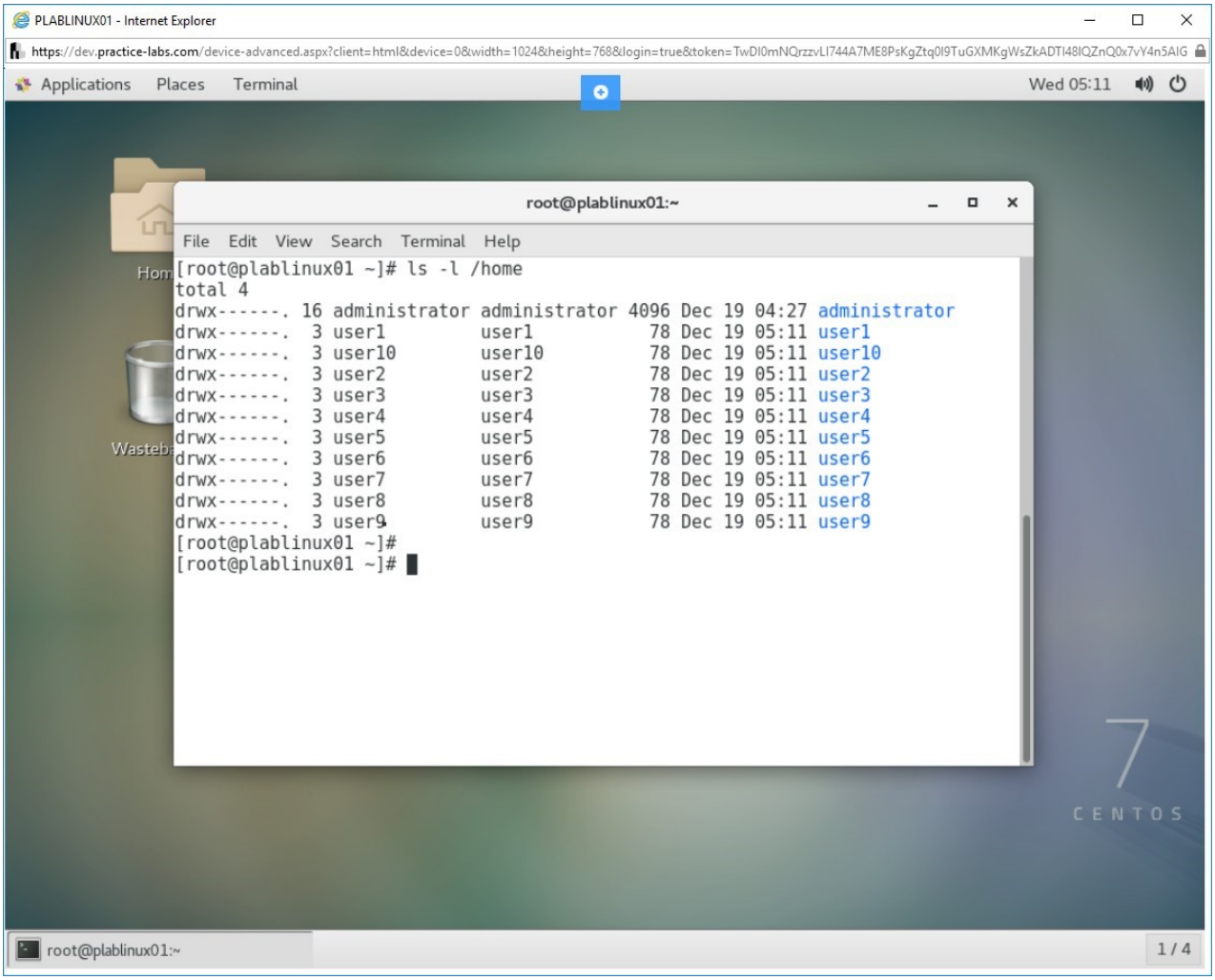


Figure 1.23 Screenshot of PLABLINUX01: Listing the contents of the /home directory.

Step 23

Clear the screen by entering the following command:

```
clear
```

Similar to the **for-do** loop, you can also write the **while** loop. It executes a list of commands repeatedly till the specified condition is reached.

Create a new script named **counter.sh** and invoke the insert mode. Type the following lines and then save the file:

```
#!/bin/bash
COUNTER=0
```

```
while [  $COUNTER -lt 10 ]
do
    echo "The counter is $COUNTER"
    (( COUNTER++ ))
done
```

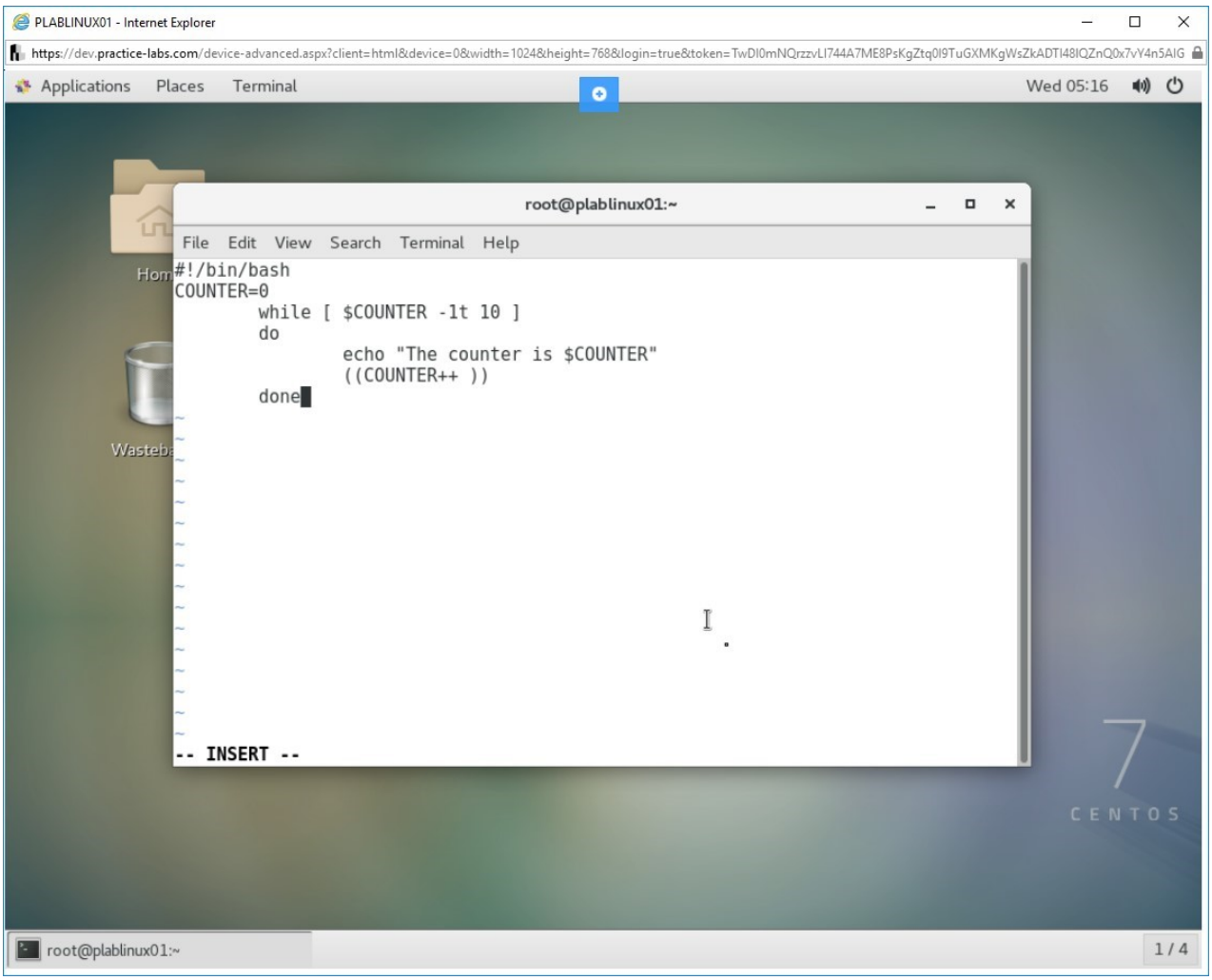


Figure 1.24 Screenshot of PLABLINUX01: Creating the script.

Step 24

Save the file by pressing **ESC** and then entering the command

```
:wq
```

Then, type the following command to make the file executable:

```
chmod +x counter.sh
```

Press **Enter**.

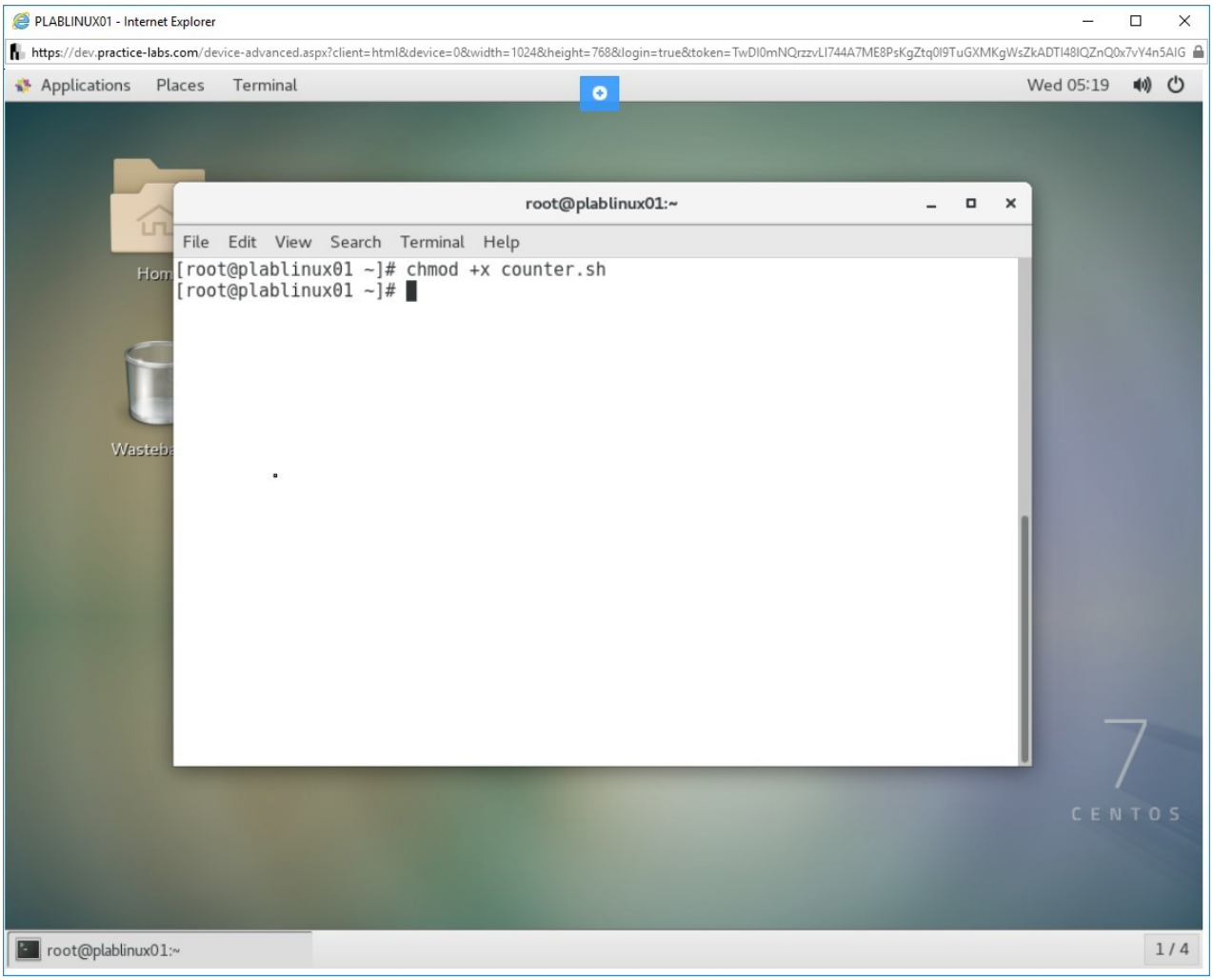


Figure 1.25 Screenshot of PLABLINUX01: Changing the permissions of the script.

Clear the screen by entering the following command:

```
clear
```

Now, run the **counter.sh** script, by typing the following command:

```
./counter.sh
```

Press **Enter**.

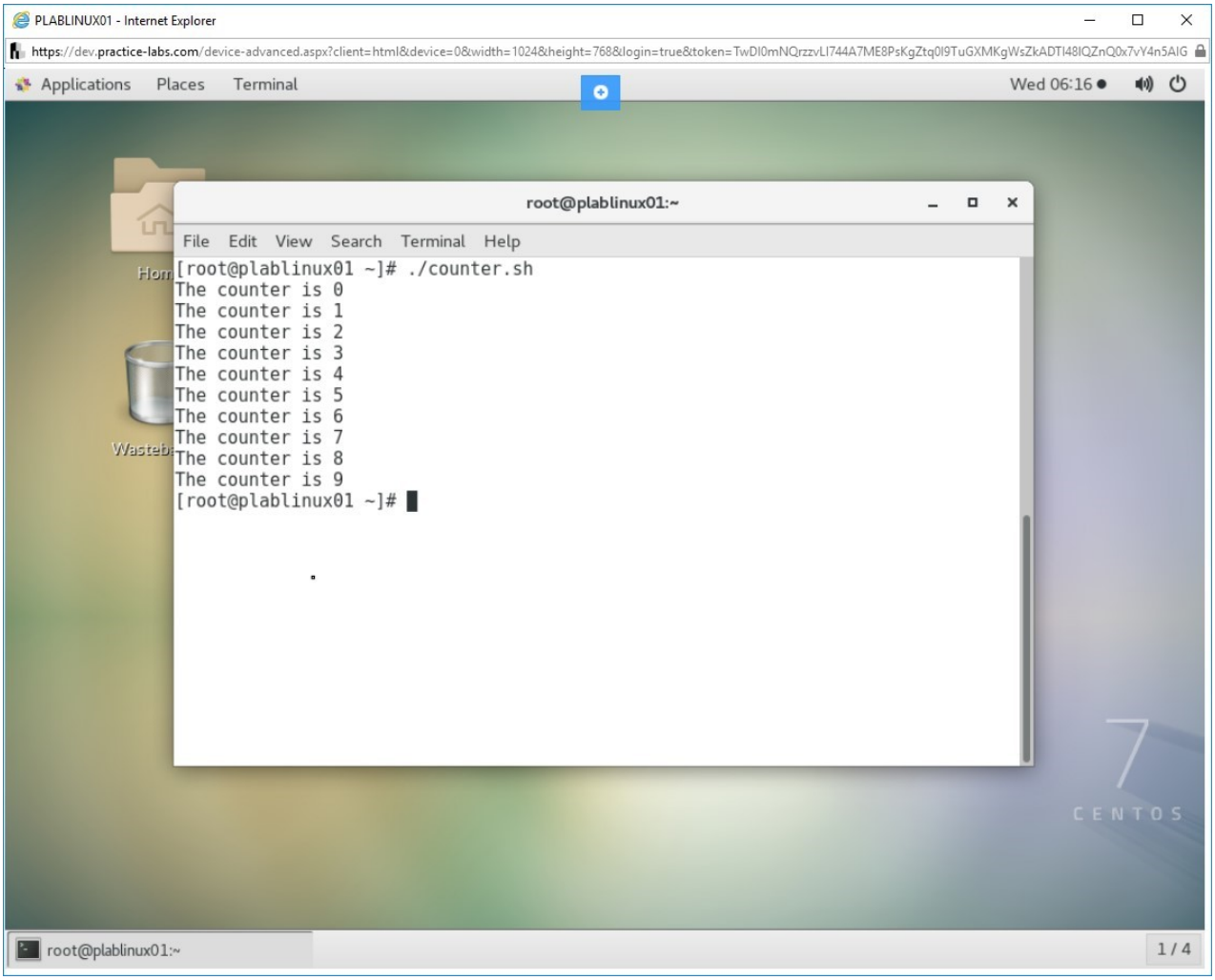


Figure 1.26 Screenshot of PLABLINUX01: Executing the script.

Step 25

Clear the screen by entering the following command:

```
clear
```

You can also use these loops without having to write a shell script. This is useful if you want to repeat a task. For example, you can define a condition in the **for** loop and then display the values on separate lines.

To use the **for** loop, type the following lines on the command prompt:

```
for planet in Mercury Venus
```

Press **Enter**.

```
do
```

Press **Enter**.

```
echo $planet
```

Press **Enter**.

```
done
```

Press **Enter**.

Note that the specified text, based on the defined condition, is displayed in separate lines.

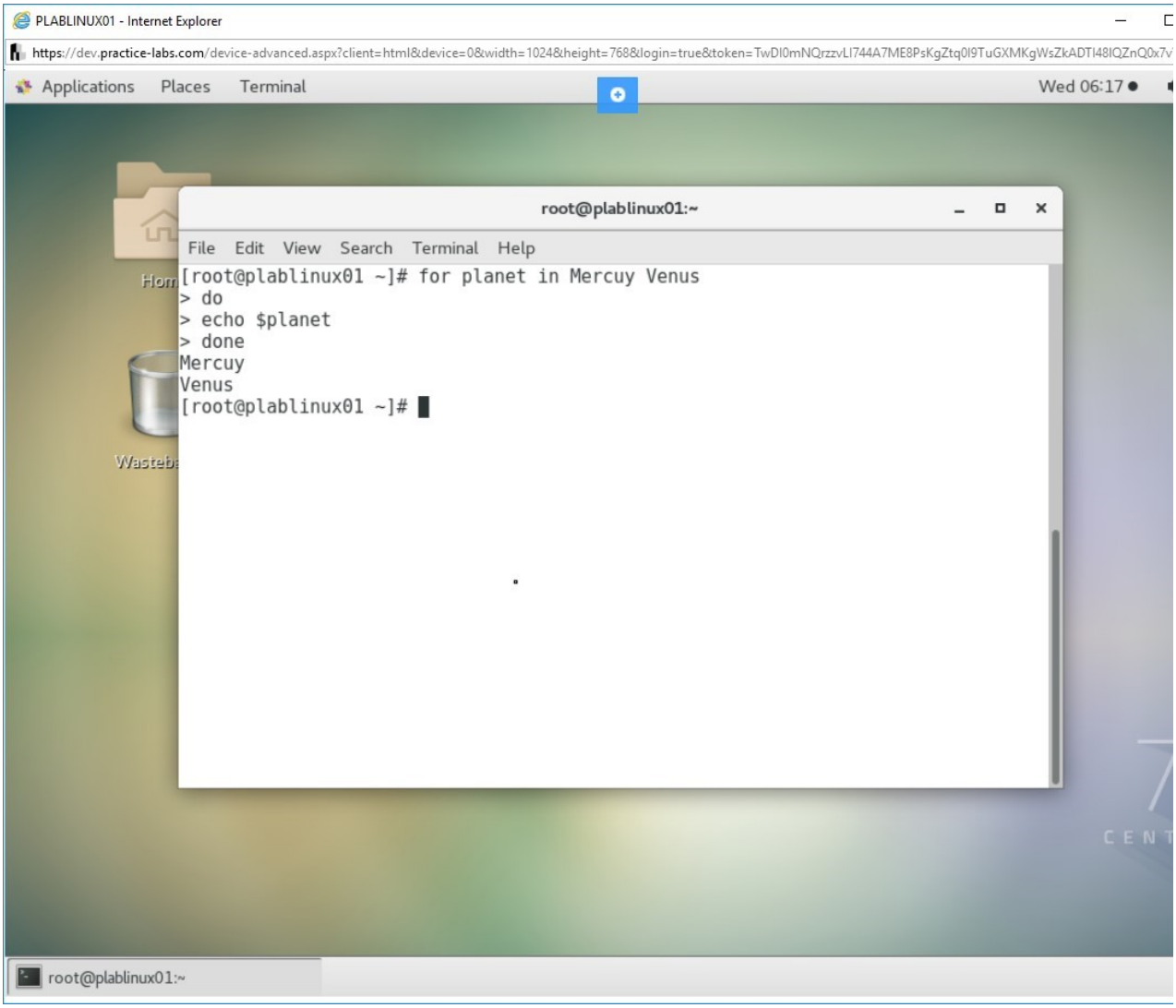


Figure 1.27 Screenshot of PLABLINUX01: Executing the for loop.

Step 26

Clear the screen by entering the following command:

```
clear
```

You can use the **if** loop to check for a condition and perform a task. If that particular condition is not met, then you can use another condition to complete a task.

Create a new script named **count.sh** and enter in the insert mode. Type the following lines and then save the file:

```
#!/bin/bash
count=99
if [ $count -eq 100 ]
then
  echo "Count is 100"
elif [ $count -gt 100 ]
then
  echo "Count is more than 100"
else
  echo "Count is less than 100"
fi
```

In this script, **if** loop is specified to have a value of **99**. It compares this value to 100 and depending on the result of the comparison, it displays the output.

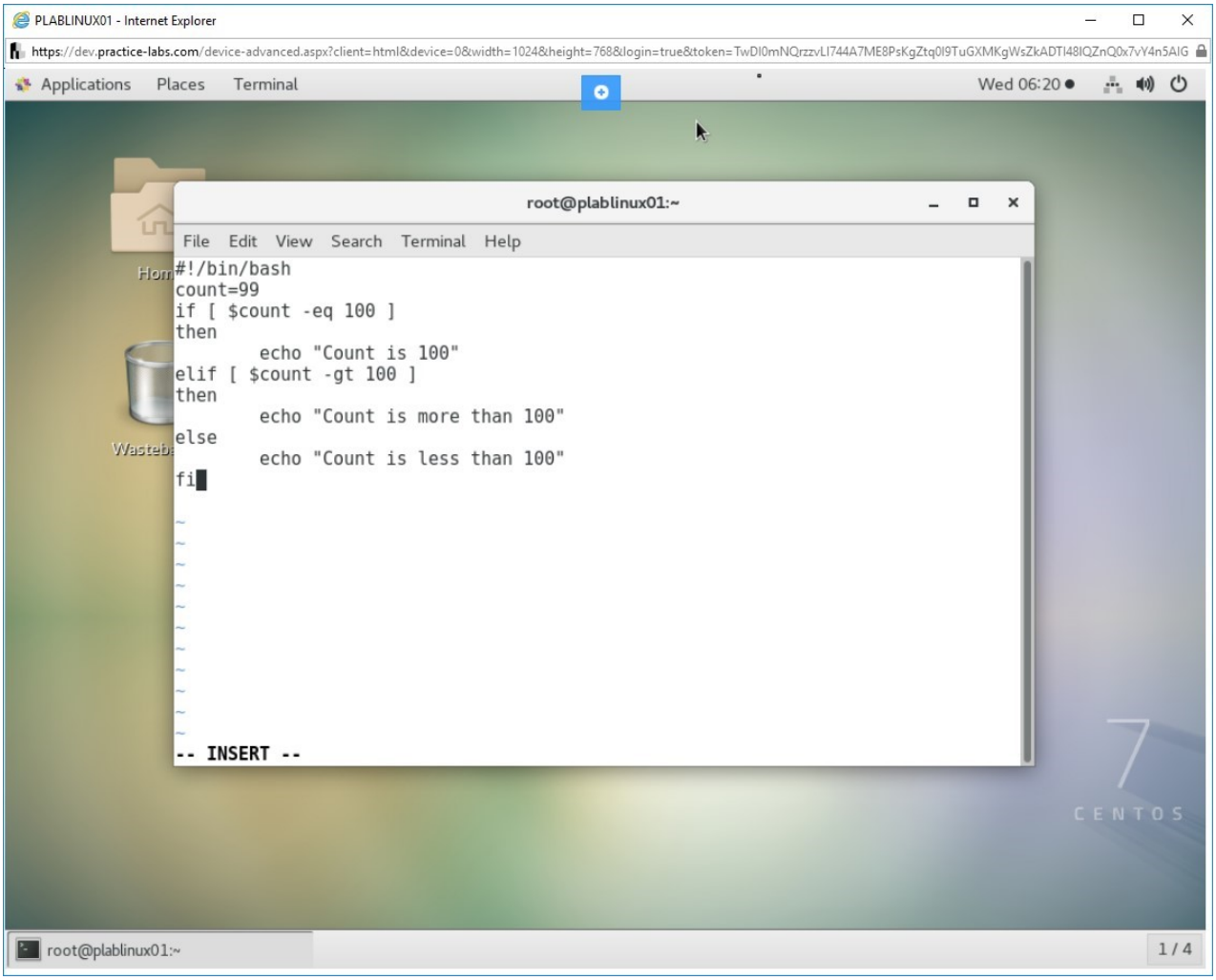


Figure 1.28 Screenshot of PLABLINUX01: Creating a script with the if loop.

Step 27

After you save the file, make it executable with the **chmod** command.

To run the **count.sh** script, type the following command:

```
./count.sh
```

Press **Enter**.

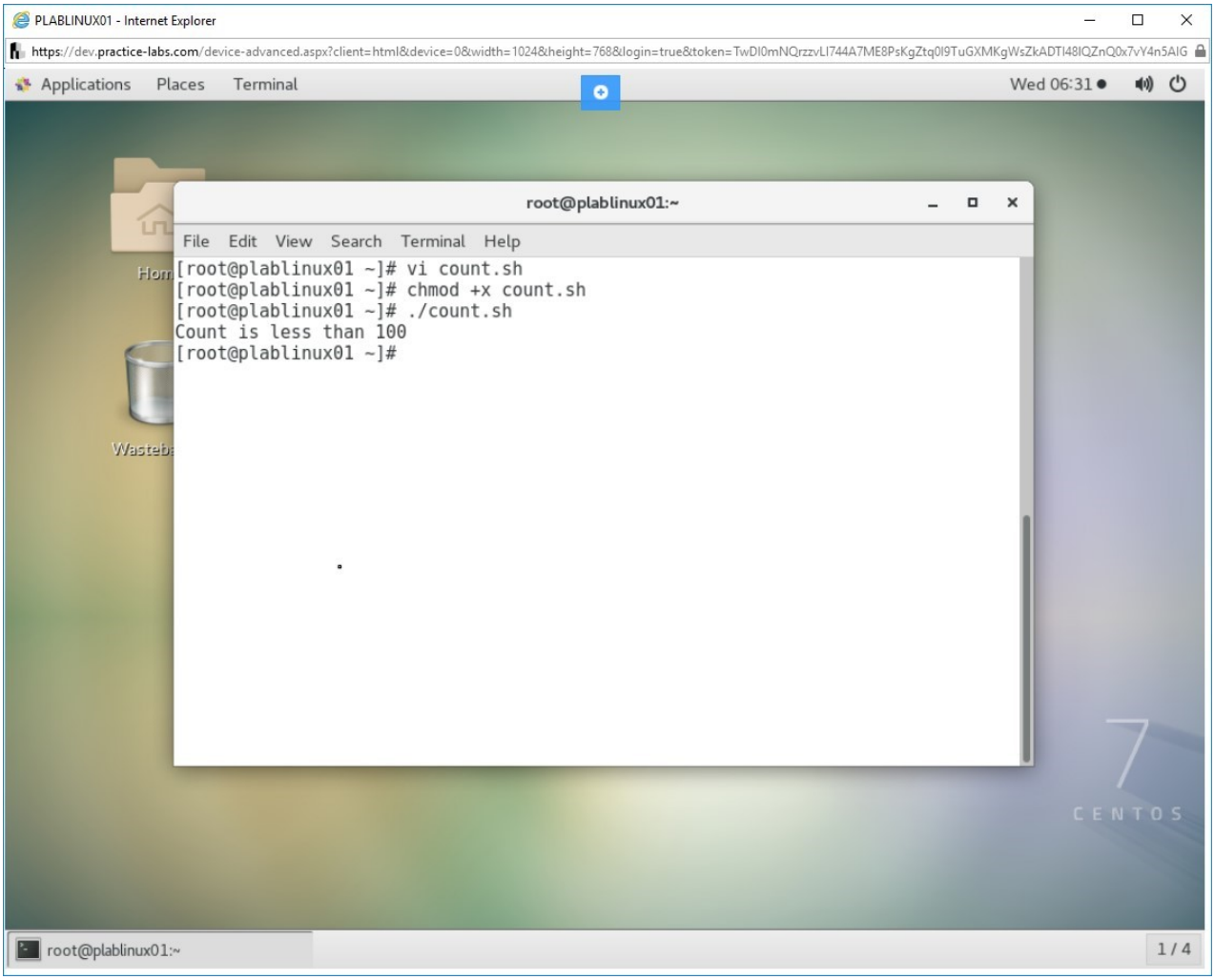


Figure 1.29 Screenshot of PLABLINUX01: Executing the script.

Task 2 - Use Command Substitution

In command substitution, a variable is substituted by its value while printing. You can assign a value to a variable. This is especially useful when a process is assigning/modifying the value of a variable, and you want to find the current value of the variable. In this task, you will use command substitution to display the value of a variable.

To use command substitution, perform the following steps:

Step 1

Clear the screen by entering the following command:

```
clear
```

You need to define a variable. To do this, type the following command:

```
var=a
```

Press **Enter**.

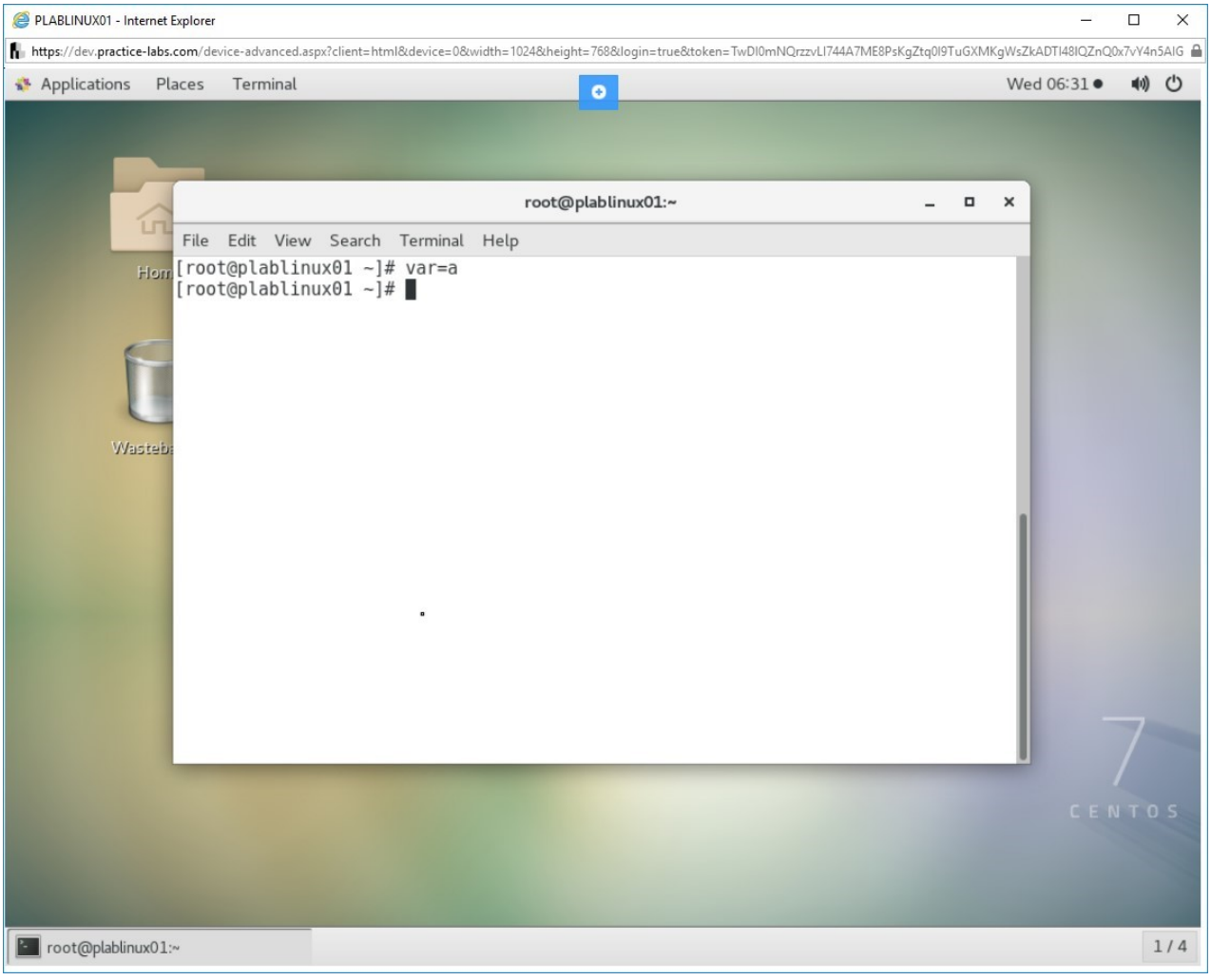


Figure 1.30 Screenshot of PLABLINUX01: Defining a variable.

Step 2

Now, you need to specify a value for this variable.

To specify the value, type the following command:

```
a=10
```

Press **Enter**.

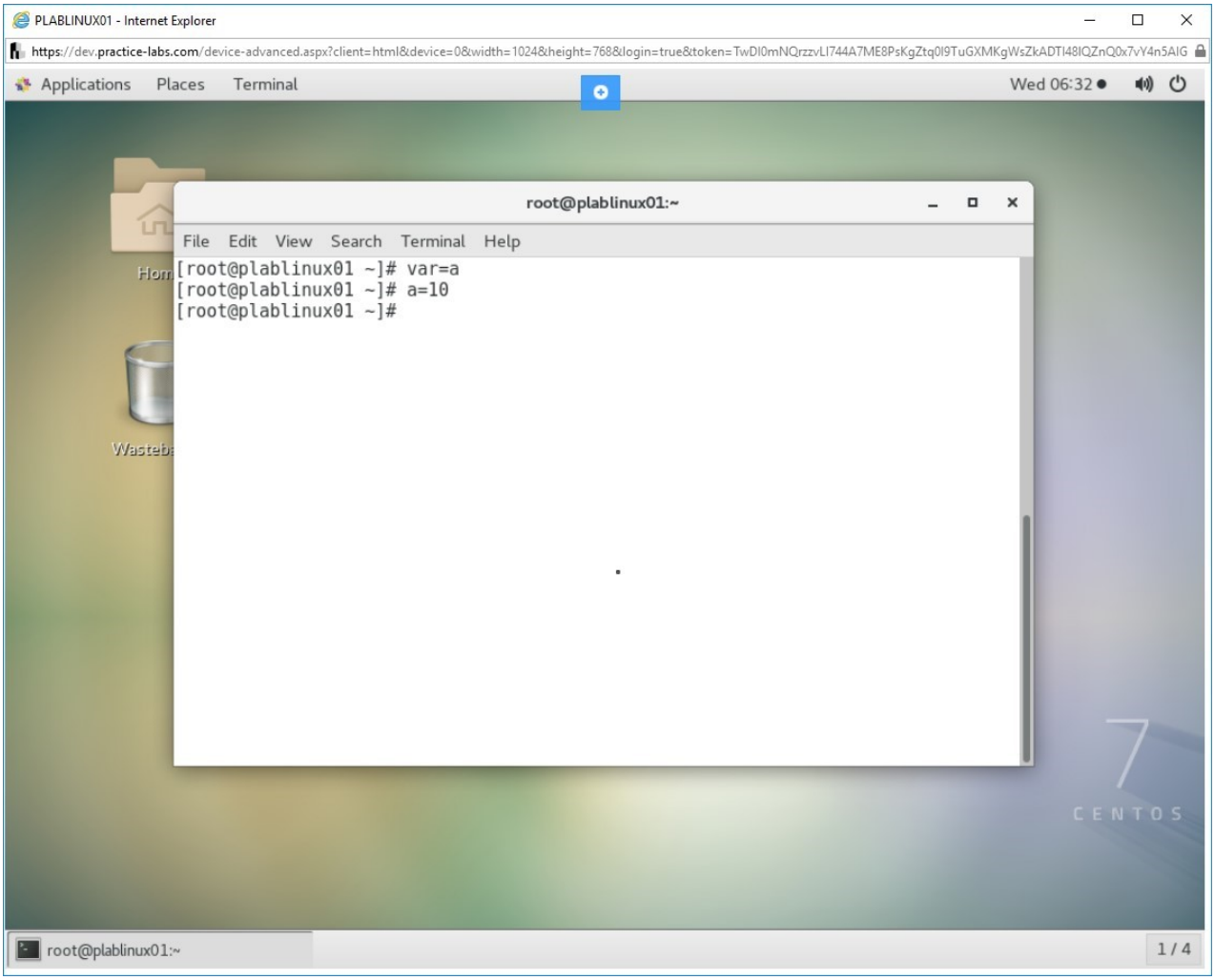


Figure 1.31 Screenshot of PLABLINUX01: Specifying a value to the variable.

Step 3

Now, you will use the variable to print its value. To do this, type the following command:

```
echo $a
```

Press **Enter**.

Notice that the value, 10, of the variable a is printed. This is a simple example of command substitution.

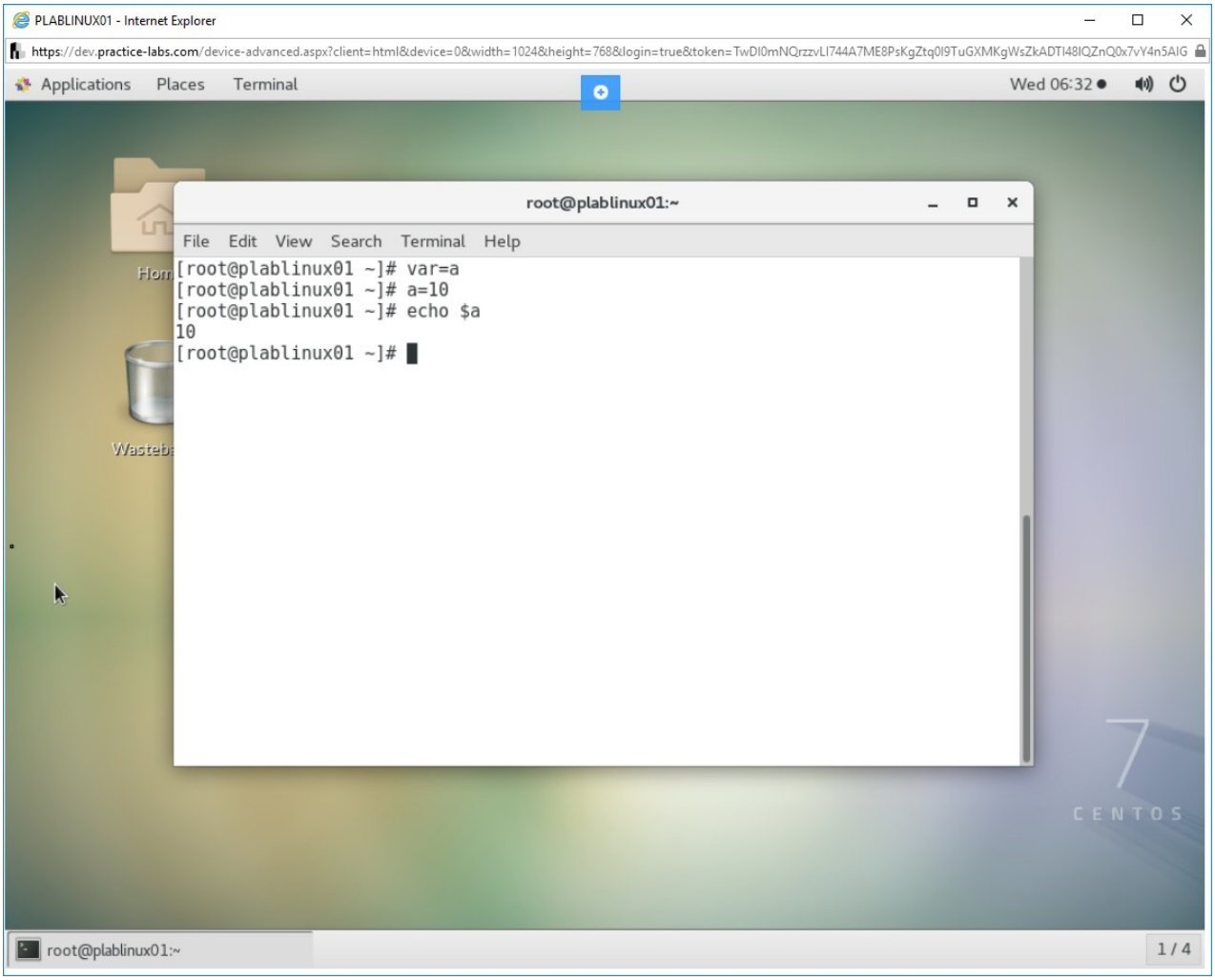


Figure 1.32 Screenshot of PLABLINUX01: Printing the value of the variable.

Step 4

Let's write a small script based on command substation. Create a new script with the name **command.sh**, invoke the insert mode, and type the following lines:

```
#! /bin/bash
DATE=$(date)
echo "Date is $DATE"
```

```
UP=$(uptime)
echo "Uptime is $UP"
```

This script will display the current date and uptime for the system.

Save the file by pressing **ESC** and the typing the command **:wq**

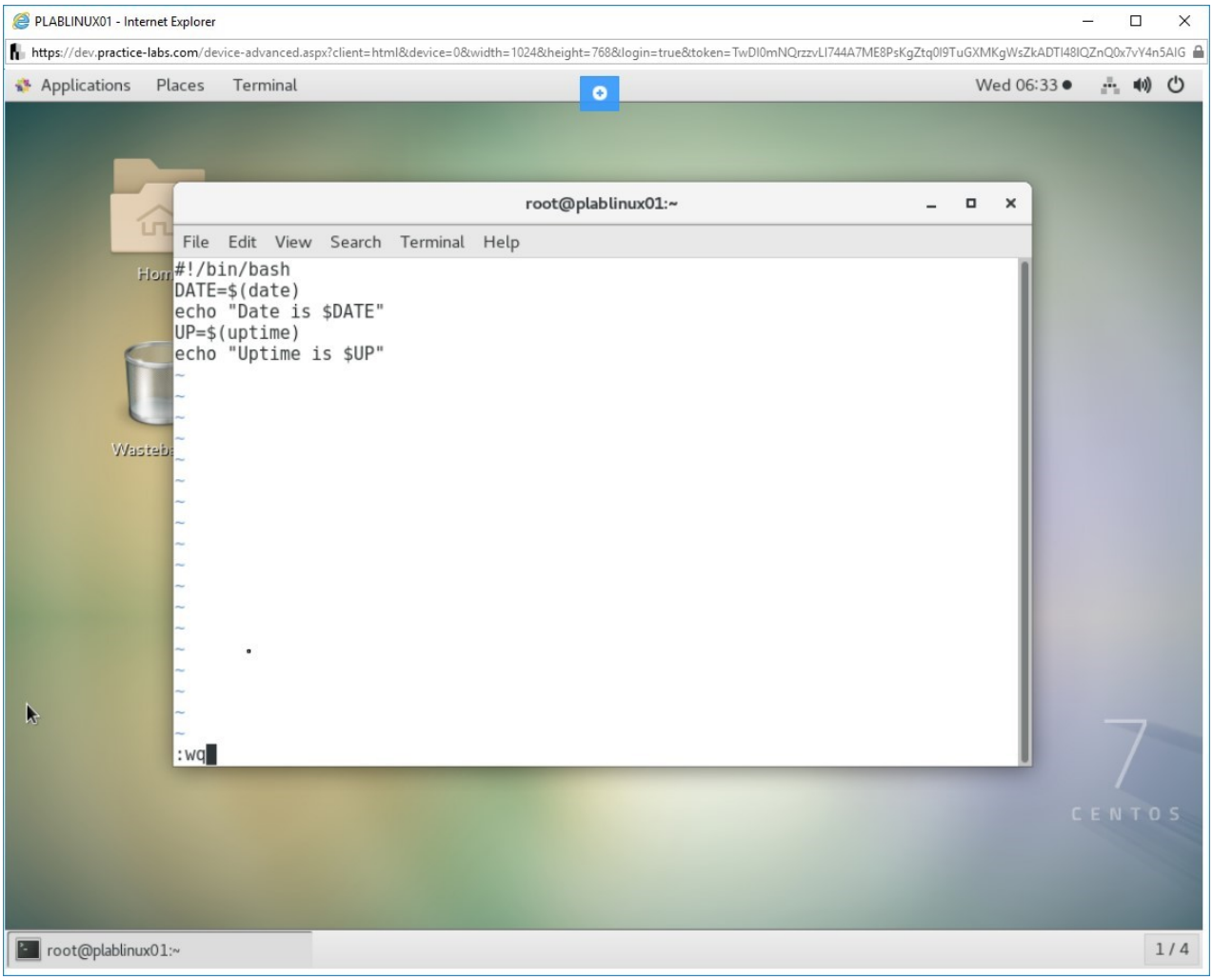


Figure 1.33 Screenshot of PLABLINUX01: Creating a script.

Step 5

Clear the screen by entering the following command:

```
clear
```

Change the permissions of the file using the **chmod** command.

To run the file, type the following command:

```
./command.sh
```

Press **Enter**.

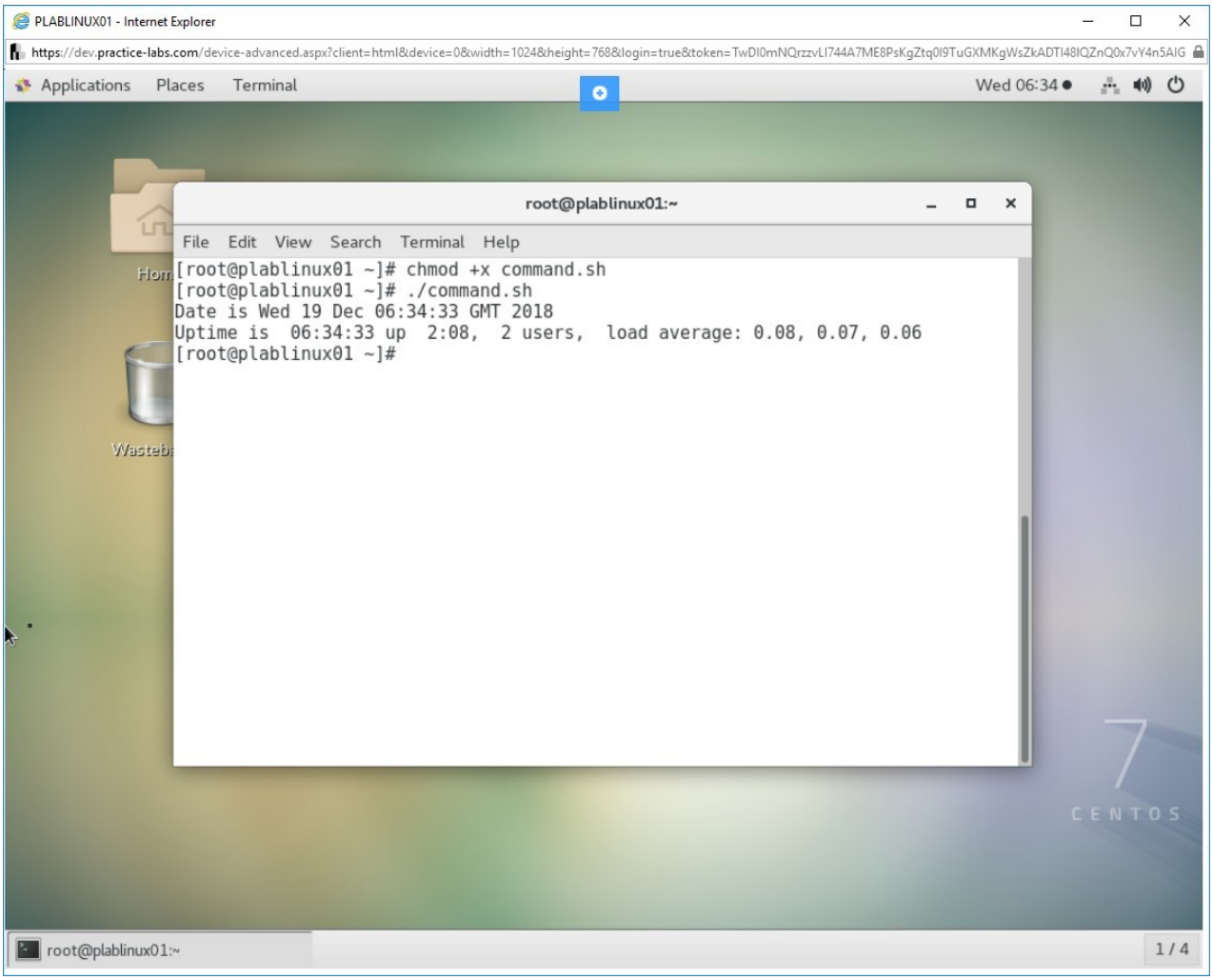


Figure 1.34 Screenshot of PLABLINUX01: Executing the script.

Task 3 - Use the Test Command

You the test command to access the value returned by a command. This value indicates success, failure, or any other status of the command. The test command can also be used for arithmetic comparisons. You can use this command for comparing the file attribute and string comparisons. In this task, you will use the test command to compare mathematical numbers. You will then use the output of this comparison to take a decision.

To use the test command, perform the following steps:

Step 1

To compare two arithmetic numbers using the test command, type the following command:

```
test 5 -gt 2 && echo "Yes"
```

Press **Enter**.

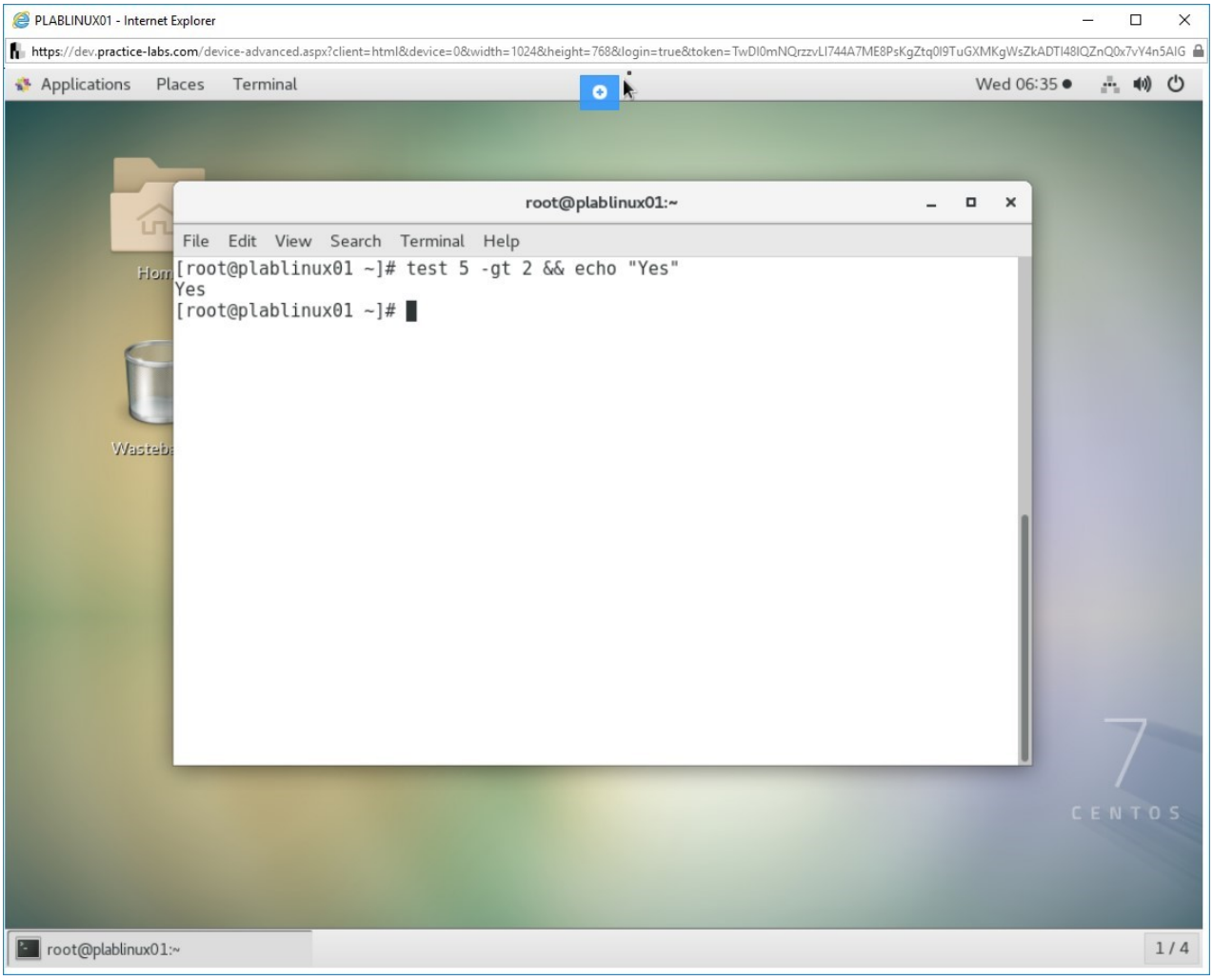


Figure 1.35 Screenshot of PLABLINUX01: Comparing two arithmetic numbers using the test command.

Step 2

You can also use **test** command to make decisions based on a condition.

To make a decision based on a condition, type the following command:

```
test 10 -eq 5 && echo Yes || echo No
```

Press **Enter**.

This command compares **10** and **5** and if they are equal, then **Yes** message is displayed. Else, **No** message is displayed.

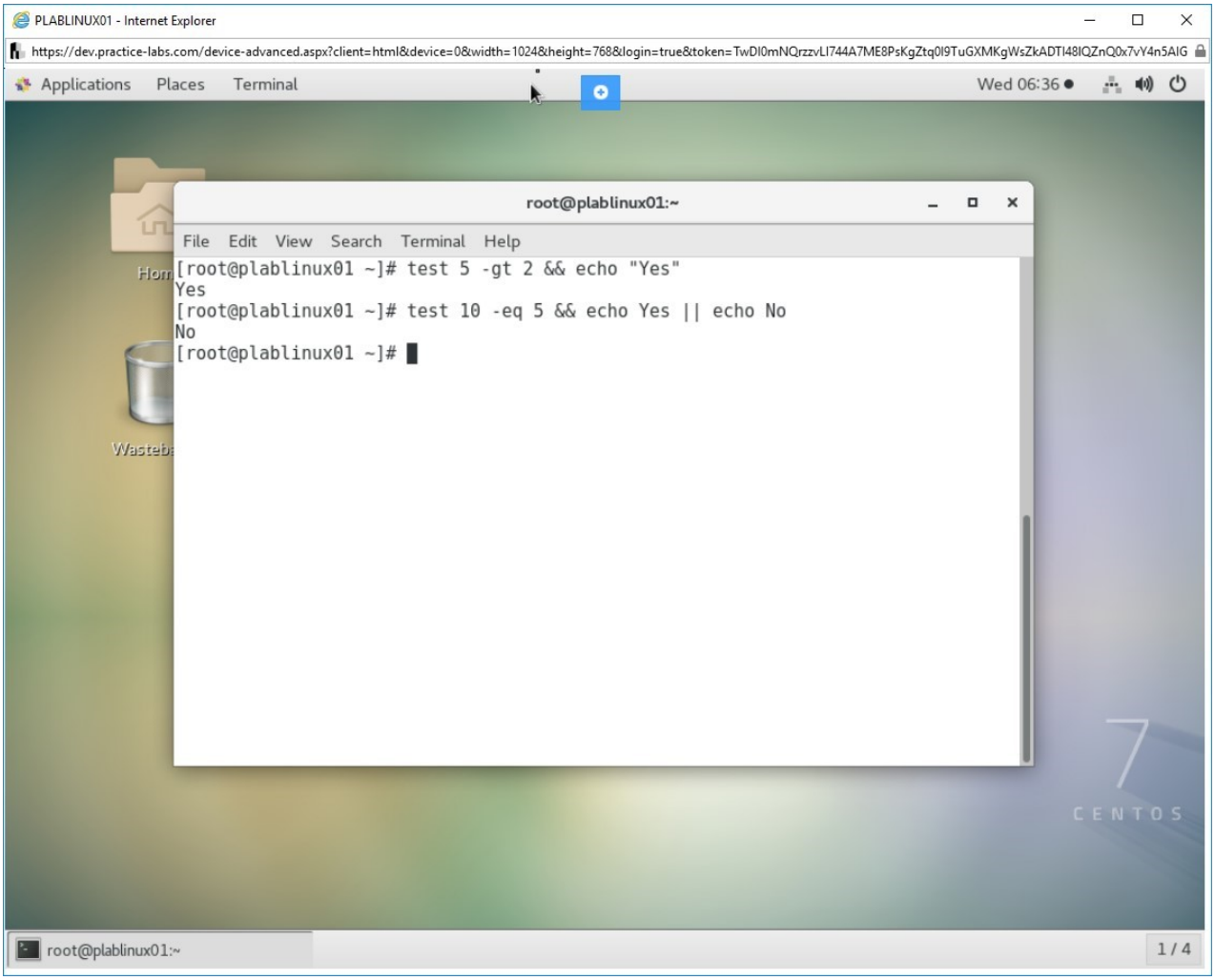


Figure 1.36 Screenshot of PLABLINUX01: Making a decision based on a condition.

Keep all devices in their current state and proceed to the next exercise.

Review

Well done, you have completed the **Customize or Write Simple Scripts** Practice Lab.

Summary

You completed the following exercise:

- Exercise 1 - Customize or Write Simple Scripts

You should now be able to:

- Use standard sh syntax
- Use command substitution
- Use the test command

Feedback

Shutdown all virtual machines used in this lab. Alternatively, you can log out of the lab platform.