Εργαστήριο 1

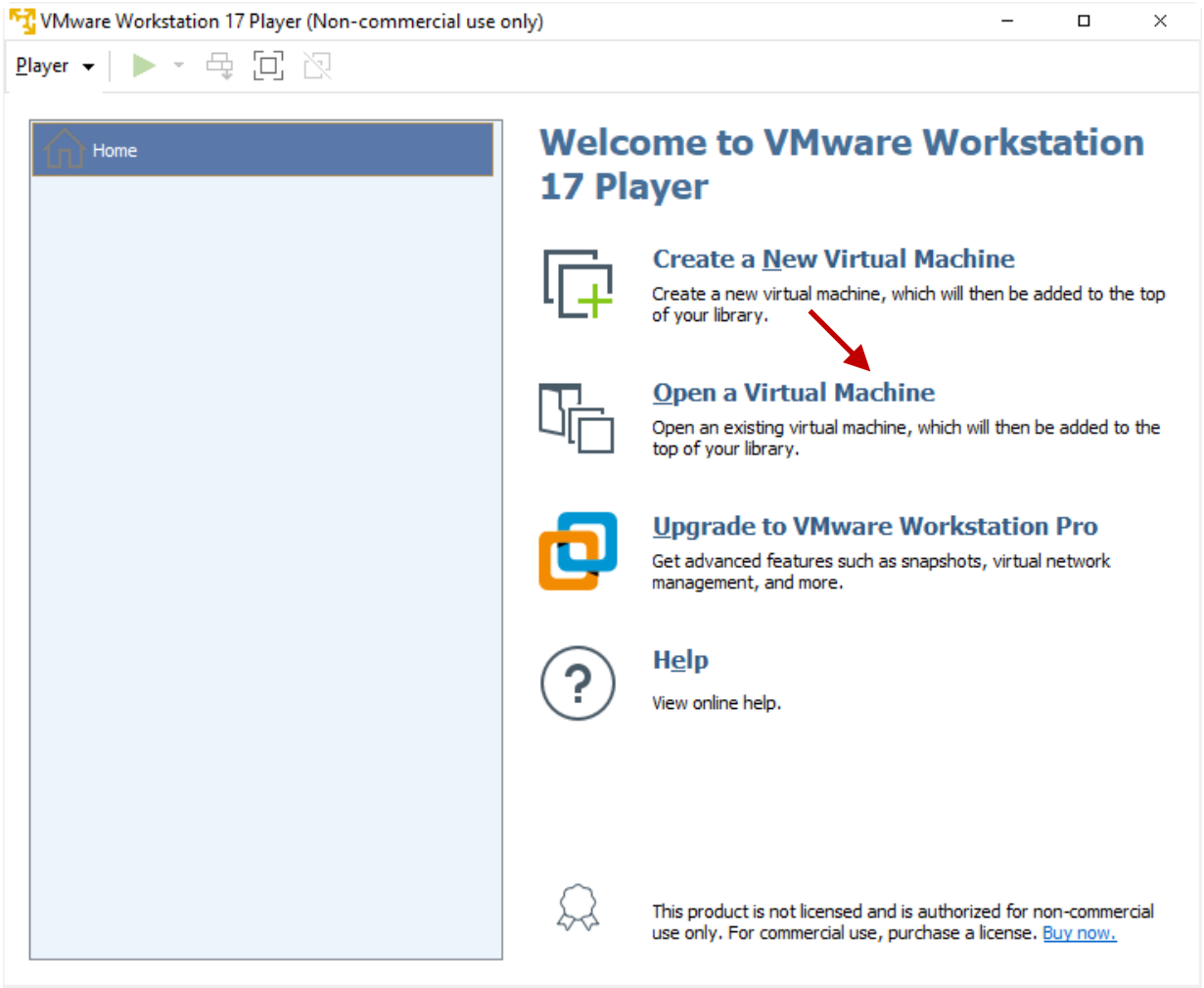
Getting started with Virtual Machines and configurations
Advanced Unix commands (grep, ps, cut, find)

1) In the context of the EPL421 lab, you are requested to install a VM image (Ubuntu 20.04 LTS 64 bit) on which you will have administrative ("root") privileges in order to install software or perform certain other functions. The VM image will be installed on a hypervisor (VMware player or Oracle Virtual Box) which you need to install in advance. Guidelines available on the lab web page. The machine on which the hypervisor is installed is called host machine.

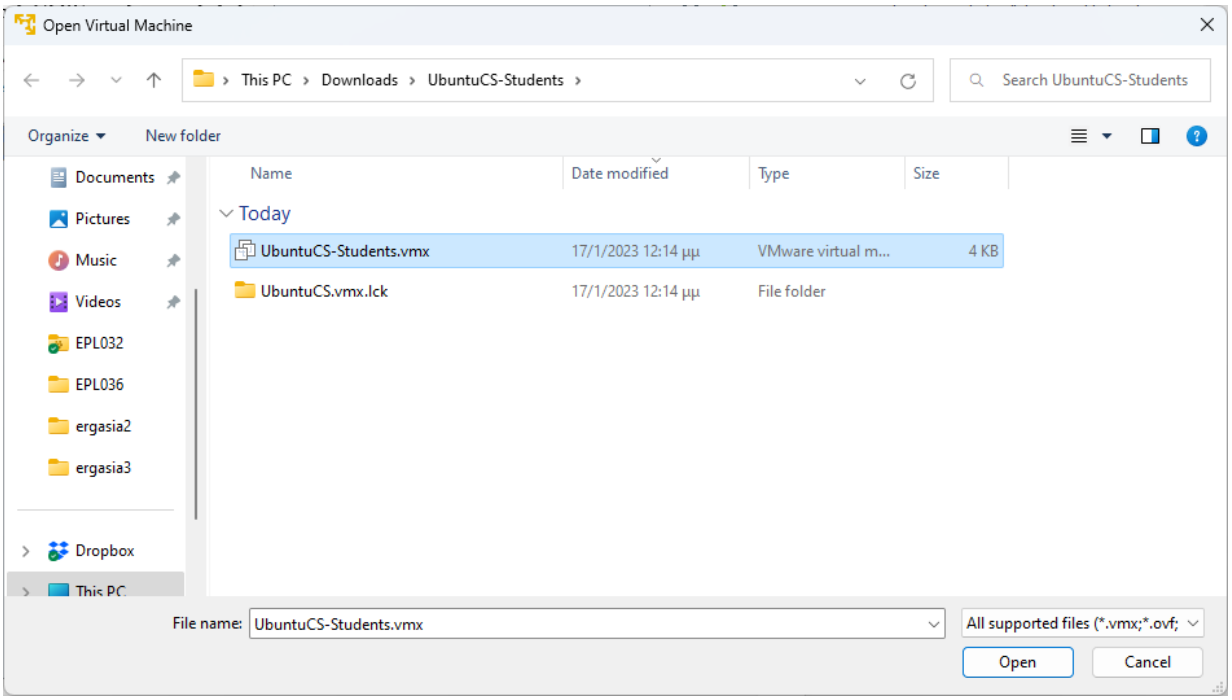
After downloading the VM image you can import it to VMWare Player or Oracle Virtual Box.

VMWare player guidelines on importing VM Ubuntu image

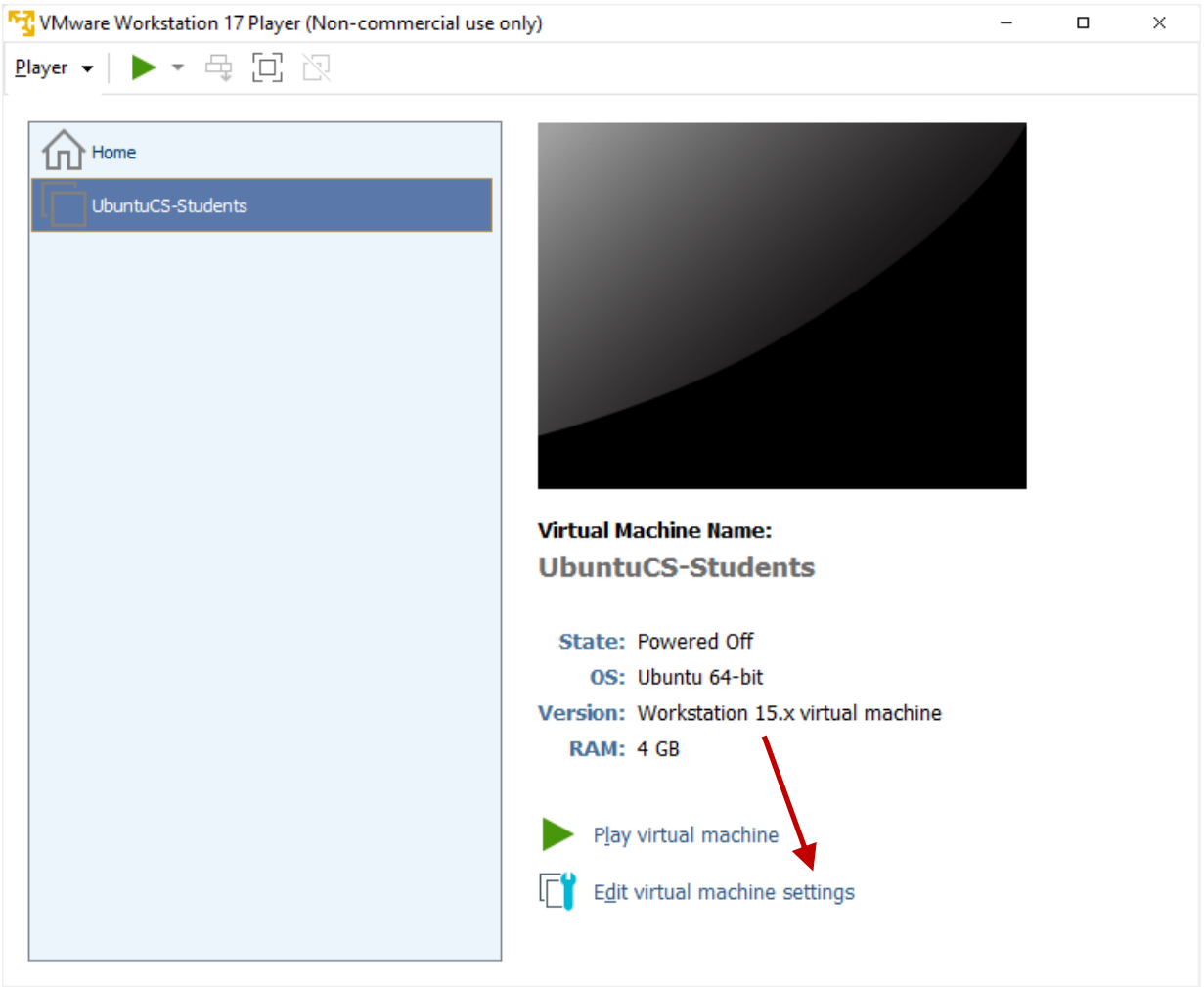
1. Download VM image and unzip it
2. Launch VMware player and click on “Open a Virtual Machine”



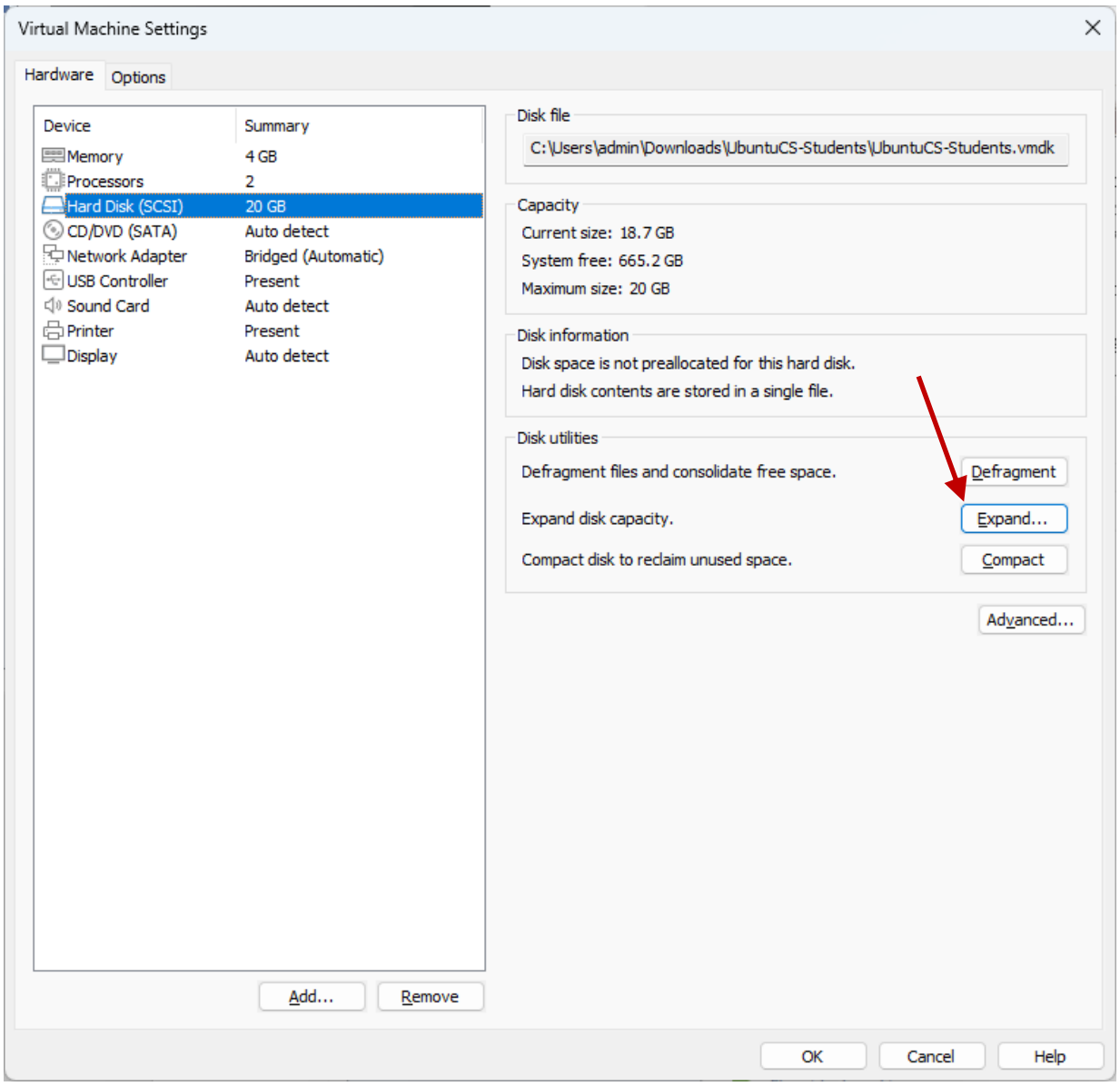
3. Navigate to the location you extracted the image and select the .vmx file

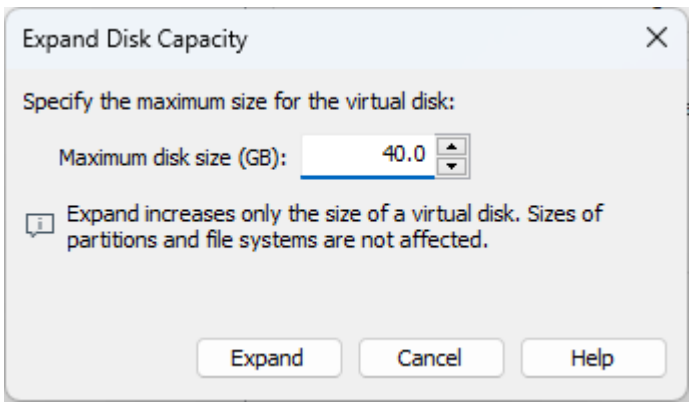


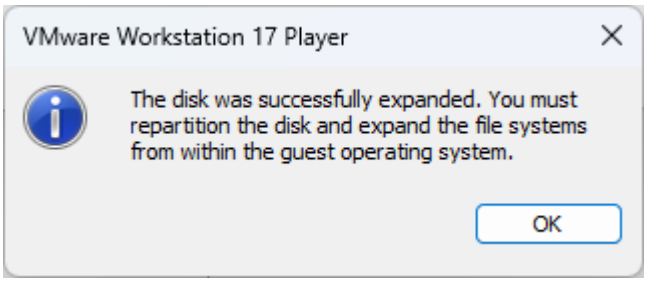
4. Click Open to import the VM to VMware player
5. Select the UbuntuCS-students VM and click on “Edit virtual machine settings”



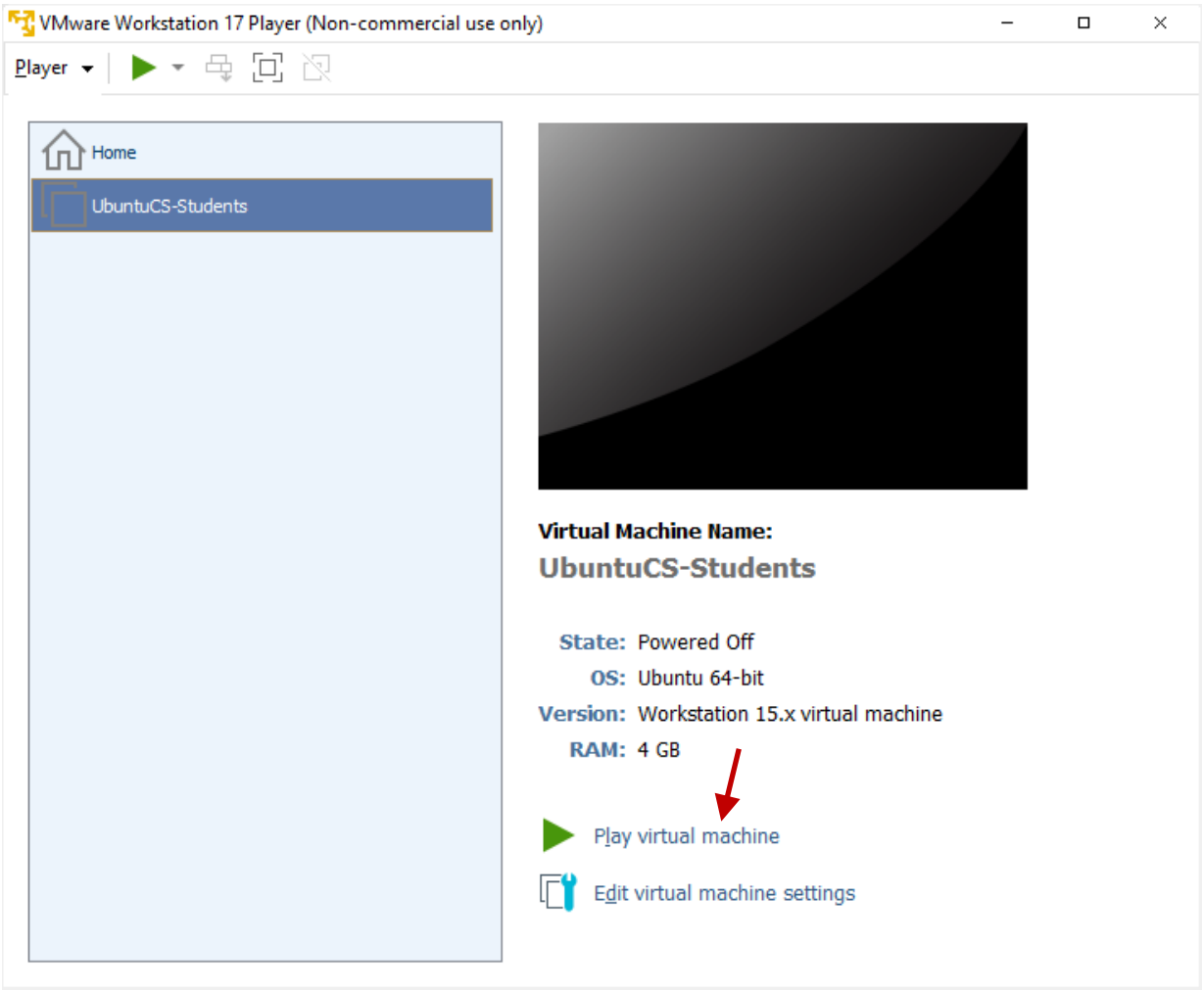
6. The VM comes with 20GB of hard disk which is rather small. If you have enough space on your host machine, I recommend to expand VM hard disk to 40GB. Double click on “Hard Disk (SCSI)” in Hardware tab and then on “Expand …” button



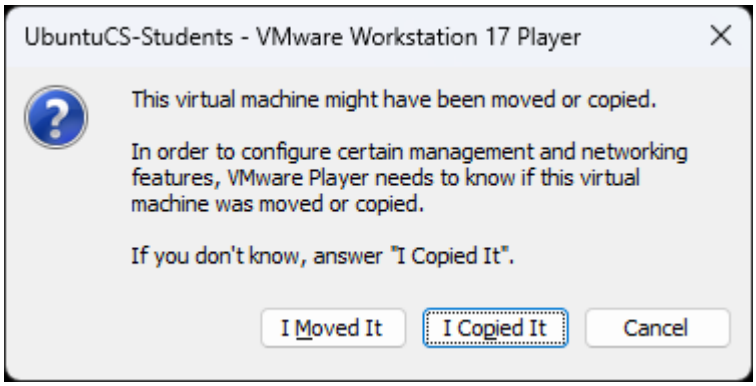




7. Power on VM

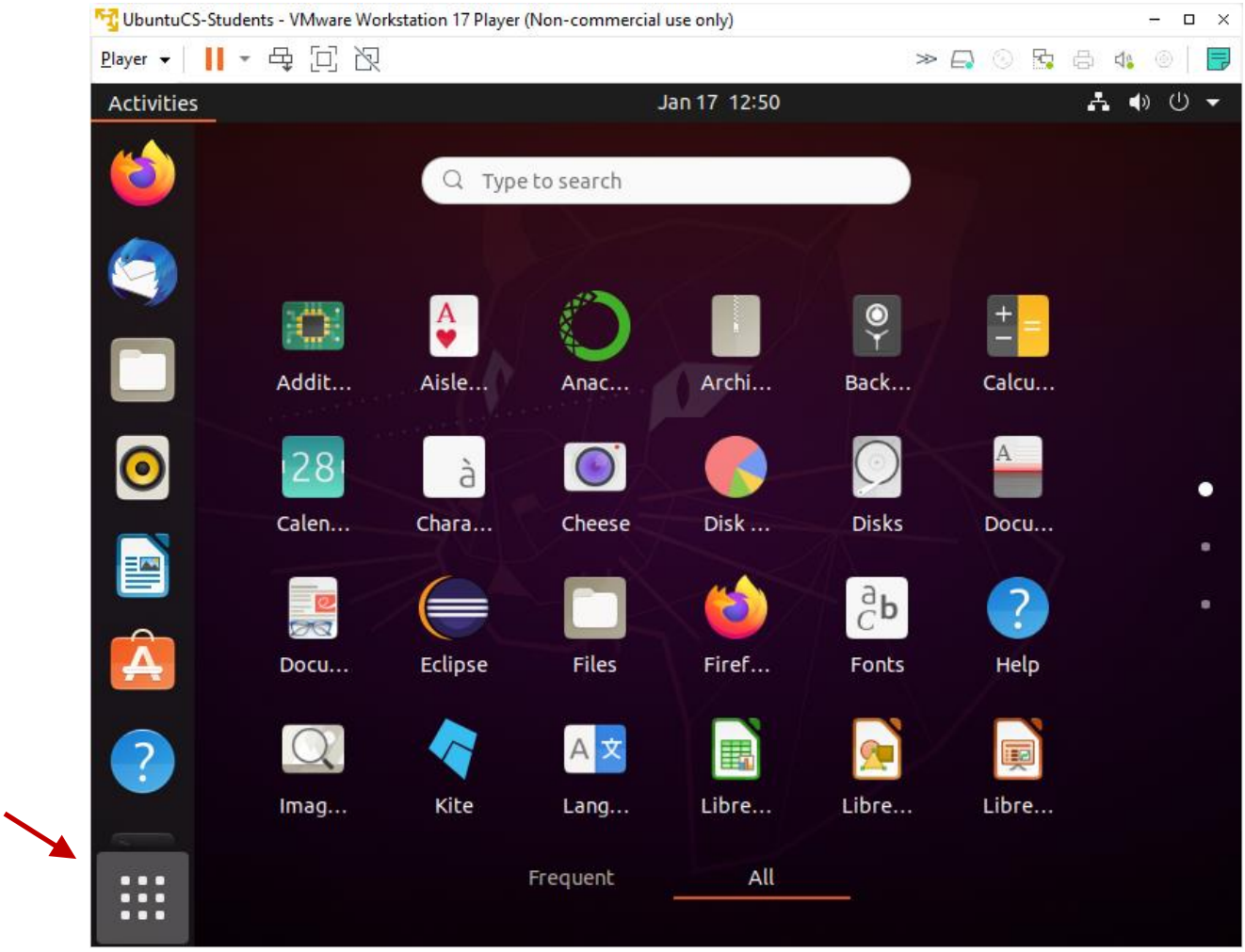


8. Click on “I copied it” on the pop up window

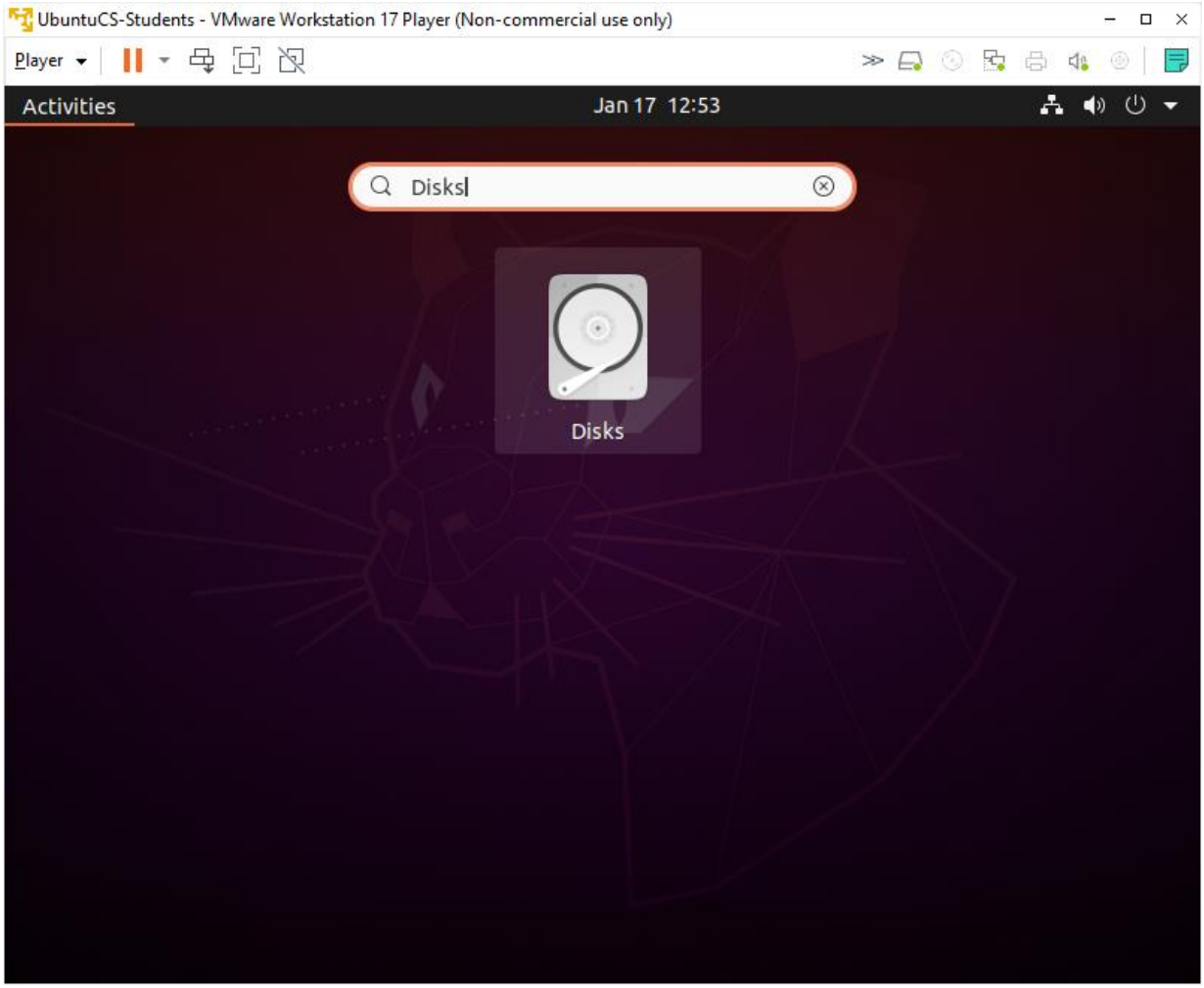


If you expanded the hard disk follow the next steps

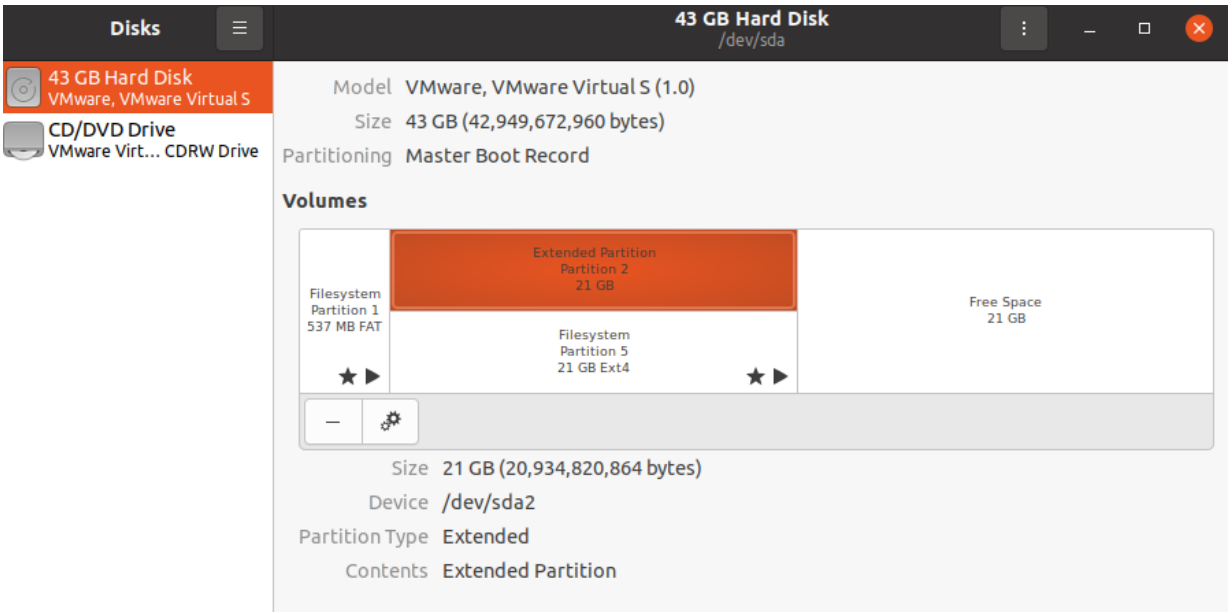
1. Open “Disks” application. This can be performed by hitting the “Show Applications” button.

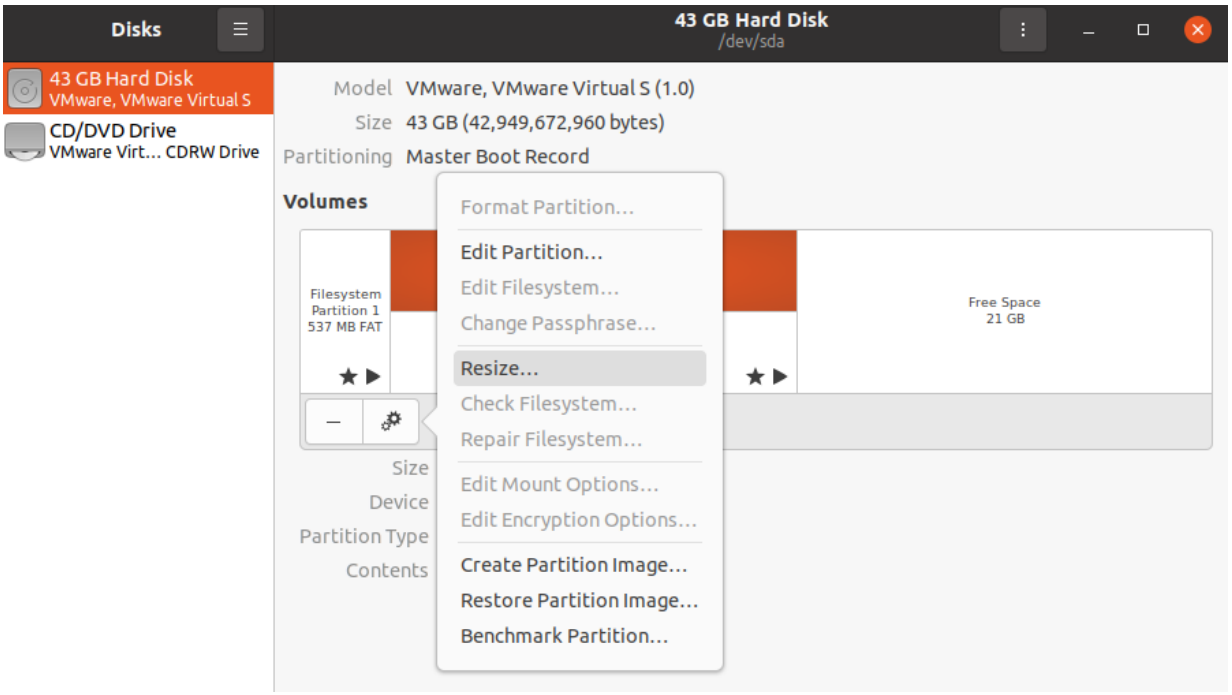


2. Type Disks and click on the Disks image

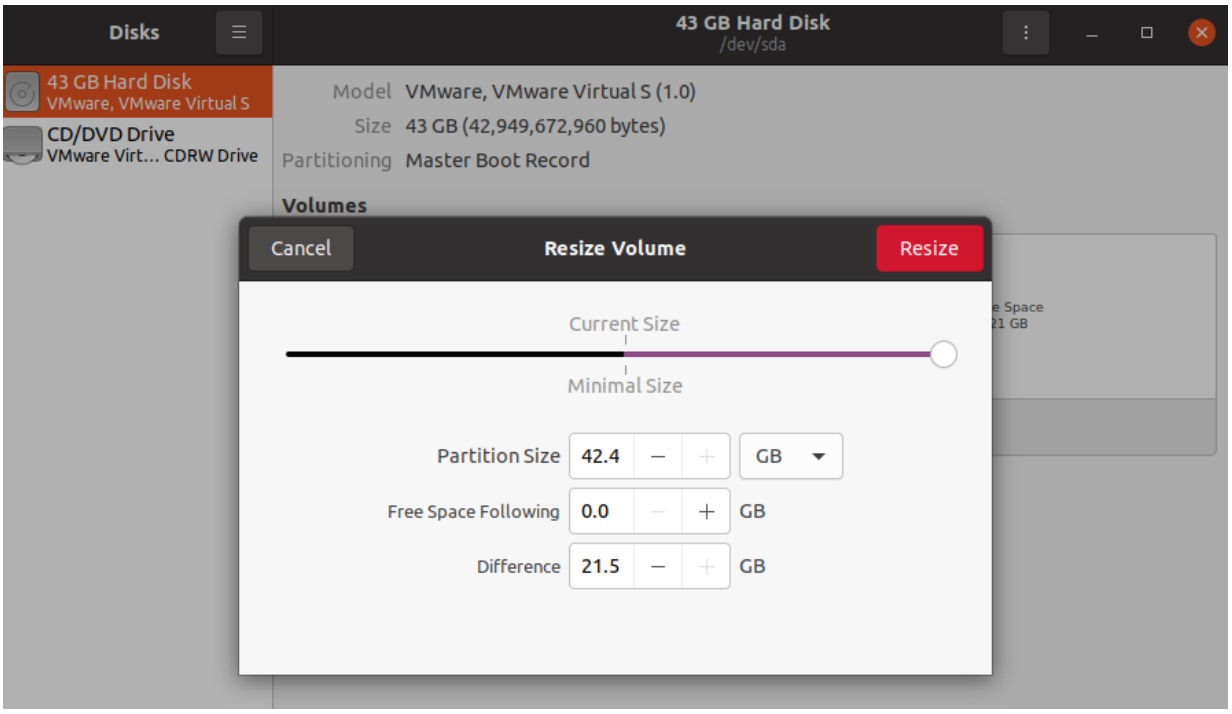


3. Click on “Extended Partition (Partition 2)” and then click on the icon with 2 cog wheels (⚙), i.e., **Additional partition options** and then **Resize**.

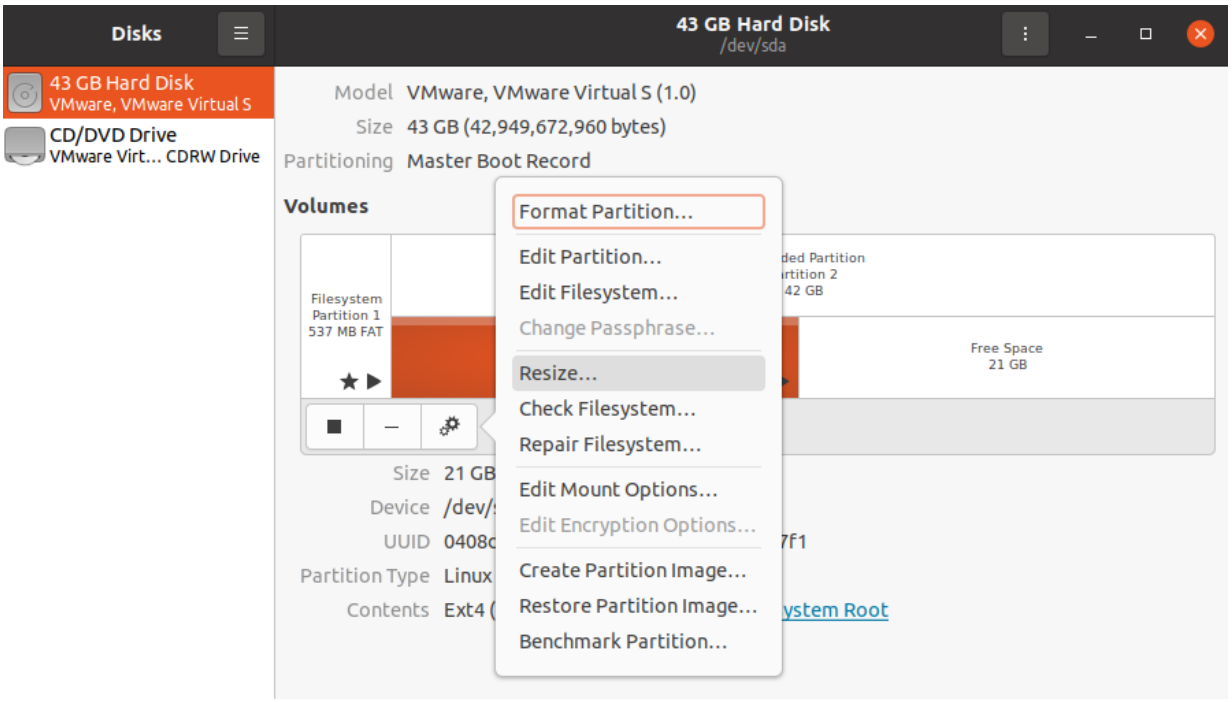




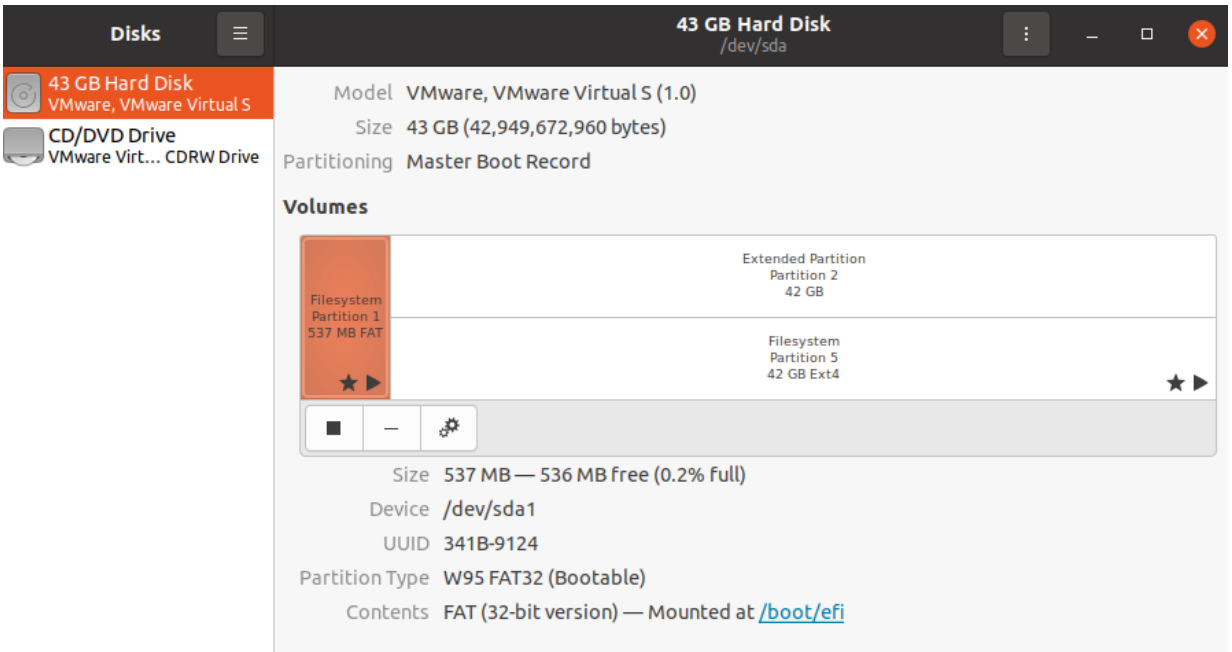
4. Drag slider to the right and click on "Resize" button.



5. Do the same for Filesystem (Partition 5)



6. Finally, you will see the following screenshot



Installation and configuration of Apache Web Server

In order to install an apache web server follow the instructions below:

A. First update the system software packages to the latest version (this may take several mins).

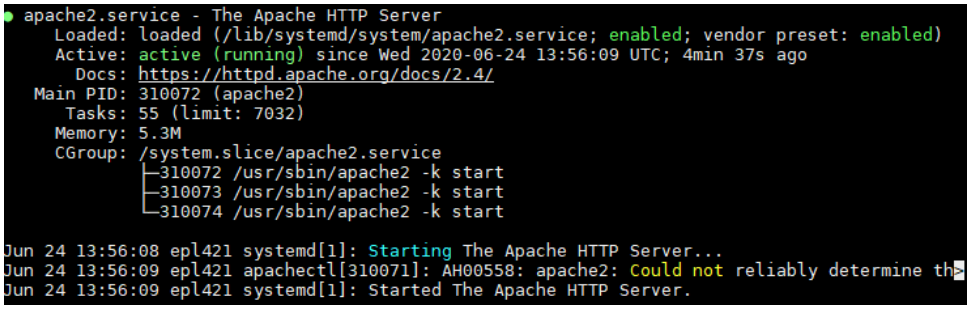
```
sudo apt update
```

B. Next, install Apache HTTP server from the default software repositories using the APT package manager as follows.

```
sudo apt install apache2
```

C. Once Apache web server installed, you can start it first time and enable it to start automatically at system boot.

```
systemctl start apache2[1]
systemctl enable apache2[2]
systemctl status apache2
```



D. Adjusting the Firewall

Before testing Apache, it's necessary to modify the firewall settings to allow outside access to the default web ports. Assuming that you followed the instructions in the prerequisites, you should have a UFW firewall configured to restrict access to your server.

During installation, Apache registers itself with UFW to provide a few application profiles that can be used to enable or disable access to Apache through the firewall.

List the ufw application profiles by typing:

```
sudo ufw app list
```

You will receive a list of the application profiles:

```
Available applications:
  Apache
  Apache Full
  Apache Secure
  CUPS
  OpenSSH
```

As indicated by the output, there are three profiles available for Apache:

- Apache: This profile opens only port 80 (normal, unencrypted web traffic)
- Apache Full: This profile opens both port 80 (normal, unencrypted web traffic) and port 443 (TLS/SSL encrypted traffic)

[1] You can use `service apache2 start`

[2] Enabling a service means it will start at boot.

- Apache Secure: This profile opens only port 443 (TLS/SSL encrypted traffic)
- CUPS (formerly an acronym for Common UNIX Printing System): This profile is related to a modular printing system which allows a computer to act as a print server. CUPS uses port 631 (TCP and UDP)
- OpenSSH: This profile opens port 22 (SSH)

It is recommended that you enable the most restrictive profile that will still allow the traffic you've configured. If you do not plan to configure SSL for your server yet you will only need to allow traffic on port 80 (no need to execute it now):

```
sudo ufw allow 'Apache'
```

In this lab, you will configure SSL for your server so you will leverage the Apache Full profile to allow both HTTP and HTTPS traffic on ports 80 and 443 respectively:

```
sudo ufw allow 'Apache Full'
```

You can verify the change by typing:

```
sudo ufw status
```

If the system outputs

```
Status: inactive
```

you have to enable firewall as shown below:

```
sudo ufw enable
```

Then you get the following prompt:

```
Command may disrupt existing ssh connections. Proceed with
operation (y|n)?
```

After hitting "y" you will see:

```
Firewall is active and enabled on system startup
```

After enabling the firewall, you can verify the previous change by typing again:

```
sudo ufw status
```

The output will provide a list of allowed HTTP and HTTPS traffic:

```
Status: active

To                         Action      From
--                         ------      ----
Apache Full                ALLOW       Anywhere
Apache Full (v6)           ALLOW       Anywhere (v6)
```

As indicated by the output, the profile has been activated to allow access to the Apache web server.
In order to be able to connect remotely to your VM using SSH you have to allow traffic on port 22 as shown below:

```
sudo ufw allow 'OpenSSH'
```

After allowing traffic on port 22, you can verify this change by typing again:

```
sudo ufw status
```

The output will provide a list of allowed HTTP and SSH traffic:

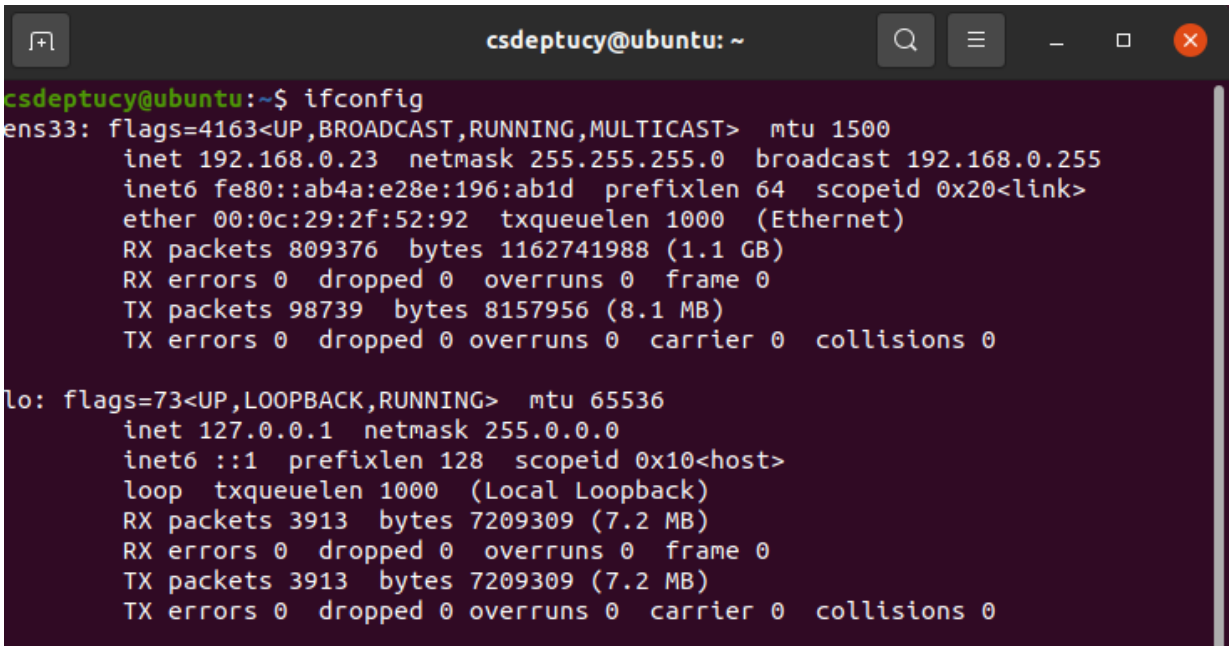
```
Status: active

To                         Action      From
--                         ------      ----
Apache Full                ALLOW       Anywhere
OpenSSH                    ALLOW       Anywhere
Apache Full (v6)           ALLOW       Anywhere (v6)
OpenSSH (v6)               ALLOW       Anywhere (v6)
```

E. Now you can verify if Apache server is up and running by typing the localhost IP address on the VM browser.

You can access the apache default homepage from your host machine (i.e Windows). Firstly you need to obtain the IP of the VM.

You can use the ifconfig command to get this information:



From the above image, it seems that the VM IP address is 192.168.0.23
So, you can open a browser from your host machine (i.e. Windows) and navigate to VM IP address. You will see the same Apache homepage as in the previous page.

F. Apache web server can be used to host more than one domains. Each domain or individual site - known as a "virtual host" - that is configured using Apache will direct the visitor to a specific directory holding that site's information. This is done without indicating that the same server is also responsible for other sites. This scheme is expandable without any software limit as long as your server can handle the load. The basic unit that describes an individual site or domain is called a `virtual host`.

Step 1 - Create the Directory Structure

The first step that we are going to take is to make a directory structure that will hold the site data that we will be serving to visitors.
Our document root (the top-level directory that Apache looks at to find content to serve) will be set to individual directories under the `/var/www/html` directory. We will create one directory here for each of the virtual hosts we plan on making.
Within each of these directories, we will create a `public_html` folder that will hold our actual files. This gives us some flexibility in our hosting.
For instance, for our sites, we're going to make our directories as follows. If you are using actual domains or alternate values, replace the highlighted text for these.

```
sudo mkdir -p /var/www/html/example1.com/public_html
sudo mkdir -p /var/www/html/test1.com/public_html
```

The highlighted portions represent the domain names that we want to serve from our VM.

Step Two - Grant Permissions

Now we have the directory structure for our files, but they are owned by our root user. If we want our regular user to be able to modify files in our web directories, we can change the ownership by doing this:

```
sudo chown -R www-data:www-data /var/www/html/example1.com/public_html
sudo chown -R www-data:www-data /var/www/html/test1.com/public_html
```

The www-data user is the apache user. By doing this, apache user now owns the `public_html` subdirectories where we will be storing our content.
We should also modify our permissions to ensure that read access is permitted to the general web directory and all of the files and folders it contains so that pages can be served correctly:

```
sudo chmod -R 755 /var/www/html/
```

Your web server should now have the permissions it needs to serve content, and your user should be able to create content within the necessary folders.

Step Three - Create Demo Pages for Each Virtual Host

We now have our directory structure in place. Let's create some content to serve.
For demonstration purposes, we'll make an index.html page for each site.
Let's begin with `example1.com`. We can open up an `index.html` file in a text editor, in this case we'll use nano:

```
sudo nano /var/www/html/example1.com/public_html/index.html
```

Within this file, create an HTML document that indicates the site it is connected to, like the following:

```
<html>
  <head>
    <title>Welcome to Example1.com!</title>
  </head>
  <body>
    <h1>Success! The example1.com virtual host is working!</h1>
  </body>
</html>
```

Save and close the file (in nano, press CTRL + X then Y then ENTER) when you are finished.
We can copy this file to use as the basis for our second site by typing:

```
sudo cp /var/www/html/example1.com/public_html/index.html
/var/www/html/test1.com/public_html/index.html
```

We can then open the file and modify the relevant pieces of information:

```
sudo nano /var/www/html/test1.com/public_html/index.html
```

```
<html>
  <head>
    <title>Welcome to Test1.com!</title>
  </head>
  <body> <h1>Success! The test1.com virtual host is
working!</h1>
  </body>
</html>
```

Save and close this file as well. You now have the pages necessary to test the virtual host configuration.

Step Four - Create New Virtual Host Files

Virtual host files are the files that specify the actual configuration of our virtual hosts and dictate how the Apache web server will respond to various domain requests.

Apache comes with a default virtual host file called `000-default.conf` that we can use as a jumping off point. We are going to copy it over to create a virtual host file for each of our domains.
We will start with one domain, configure it, copy it for our second domain, and then make the few further adjustments needed. The default Ubuntu configuration requires that each virtual host file end in `.conf`.

Create the First Virtual Host File

Start by copying the file for the first domain:

```
sudo cp /etc/apache2/sites-available/000-default.conf
/etc/apache2/sites-available/example1.com.conf
```

Open the new file in your editor with root privileges:

```
sudo nano /etc/apache2/sites-available/example1.com.conf
```

With comments removed, the file will look similar to this:

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Within this file, we will customize the items for our first domain and add some additional directives. This virtual host section matches *any* requests that are made on port 80, the default HTTP port.
First, we need to change the ServerAdmin directive to an email that the site administrator can receive emails through.

```
ServerAdmin admin@example1.com
```

After this, we need to ***add*** two directives. The first, called `ServerName`, establishes the base domain that should match for this virtual host definition. This will most likely be your domain. The second, called `ServerAlias`, defines further names that should match as if they were the base name. This is useful for matching hosts you defined, like www:

```
ServerName example1.com
ServerAlias www.example1.com
```

The only other thing we need to change for our virtual host file is the location of the document root for this domain. We already created the directory we need, so we just need to alter the `DocumentRoot` directive to reflect the directory we created:

```
DocumentRoot /var/www/html/example1.com/public_html
```

When complete, our virtual host file should look like this:

```
<VirtualHost *:80>
    ServerAdmin admin@example1.com
    ServerName example1.com
    ServerAlias www.example1.com
    DocumentRoot /var/www/html/example1.com/public_html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

At this point, save and close the file.

Copy First Virtual Host and Customize for Second Domain

Now that we have our first virtual host file established, we can create our second one by copying that file and adjusting it as needed. Start by copying it:

```
sudo cp /etc/apache2/sites-available/example1.com.conf
/etc/apache2/sites-available/test1.com.conf
```

Open the new file with root privileges in your editor:

```
sudo nano /etc/apache2/sites-available/test1.com.conf
```

You now need to modify all of the pieces of information to reference your second domain. When you are finished, it should look like this:

```
<VirtualHost *:80>
    ServerAdmin admin@test1.com
    ServerName test1.com
    ServerAlias www.test1.com
    DocumentRoot /var/www/html/test1.com/public_html
    ErrorLog ${APACHE_LOG_DIR}/error.log
```

```
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Save and close the file when you are finished.

Step Five - Enable the New Virtual Host Files

Now that we have created our virtual host files, we must enable them. Apache includes some tools that allow us to do this.
We'll be using the `a2ensite` tool to enable each of our sites. If you would like to read more about this script, you can refer to the a2ensite documentation.

```
sudo a2ensite example1.com.conf
sudo a2ensite test1.com.conf
```

When you are finished, you need to restart Apache to make these changes take effect and use systemctl status to verify the success of the restart.

```
sudo systemctl restart apache2
sudo systemctl status apache2
```
Your server should now be set up to serve two websites.

Step Six - Set Up Local Hosts File

If you haven't been using actual domain names that you own to test this procedure and have been using some example domains instead (i.e. example1.com and test1.com in our case), you can at least test the functionality of this process by temporarily modifying the hosts file on the virtual machine.

This will intercept any requests for the domains that you configured and point them to your VM, just as the DNS system would do if you were using registered domains. This will only work from your VM though, and only for testing purposes.

Edit your hosts file with administrative privileges by typing:

```
sudo nano /etc/hosts
```

The details that you need to add are the public IP address of your virtual machine followed by the domain you want to use to reach. Using the domains used in this guide, and replacing your server IP for your_VM_server_IP string, your file should look like this:

```
127.0.0.1   localhost
127.0.1.1   guest-desktop
your_VM_server_IP example1.com
your_VM_server_IP test1.com
your_VM_server_IP www.example1.com
your_VM_server_IP www.test1.com
```

Save and close the file.
This will direct any requests for example1.com and test1.com:

to the apache server on the virtual machine. This is what we want if we are not actually the owners of these domains in order to test our virtual hosts.

G. Secure Apache with Let's Encrypt

Step One - Installing Certbot

In order to obtain an SSL certificate with Let's Encrypt, we'll first need to install the Certbot software on your server. We'll use the default Ubuntu package repositories for that.

We need two packages: certbot, and python3-certbot-apache. The latter is a plugin that integrates Certbot with Apache, making it possible to automate obtaining a certificate and configuring HTTPS within your web server with a single command.

```
sudo apt install certbot python3-certbot-apache
```

You will be prompted to confirm the installation by pressing Y, then ENTER.

Certbot is now installed on your server.

Step Two - Obtaining an SSL Certificate

Certbot provides a variety of ways to obtain SSL certificates through plugins. The Apache plugin will take care of reconfiguring Apache and reloading the configuration whenever necessary. To use this plugin, type the following:

```
sudo certbot --apache
```

This script will prompt you to answer a series of questions in order to configure your SSL certificate. First, it will ask you for a valid e-mail address. This email will be used for renewal notifications and security notices:

```
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator apache, Installer apache
Enter email address (used for urgent renewal and security
notices) (Enter 'c' to
cancel): your_username@ucy.ac.cy
```

After providing a valid e-mail address, hit ENTER to proceed to the next step. You will then be prompted to confirm if you agree to Let's Encrypt terms of service. You can confirm by pressing A and then ENTER:

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.3-September-21-2022.pdf. You must
agree in order to register with the ACME server at
https://acme-v02.api.letsencrypt.org/directory
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(A)gree/(C)ancel: A
```

Next, you'll be asked if you would like to share your email with the Electronic Frontier Foundation to receive news and other information. If you do not want to subscribe to their content, type N. Otherwise, type Y. Then, hit ENTER to proceed to the next step.

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you be willing to share your email address with the Electronic Frontier
Foundation, a founding partner of the Let's Encrypt project and the non-profit
organization that develops Certbot? We'd like to send you email about our work
encrypting the web, EFF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: N
```

The next step will prompt you to inform Certbot of which domains you'd like to activate HTTPS for. The listed domain names are automatically obtained from your Apache virtual host configuration, that's why it's important to make sure you have the correct ServerName and ServerAlias settings configured in your virtual host. If you'd like to enable HTTPS for all listed domain names (recommended), you can leave the prompt blank and hit ENTER to proceed. Otherwise, select the domains you want to enable HTTPS for by listing each appropriate number, separated by commas and/ or spaces, then hit ENTER.

```
Which names would you like to activate HTTPS for?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: example1.com
2: www.example1.com
3: test1.com
4. www.test1.com
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate numbers separated by commas and/or spaces, or leave input
blank to select all options shown (Enter 'c' to cancel):
```

At this point, the certbot will try to validate the domain names for which you want to enable HTTPS by issuing requests to the DNS server. You cannot proceed beyond that point since the domain names (example1.com and test1.com) are not registered in a public domain name registry. In case you have a public domain name and want to proceed further please refer to this tutorial.

In the next steps, the certificate for each domain name will be created and virtual host files will be added at /etc/apache2/sites-available/your_domain-le-ssl.conf. The Apache SSL module will be enabled. Next, you'll be prompted to select whether or not you want HTTP traffic redirected to HTTPS. In practice, that means when someone visits your website through unencrypted channels (HTTP), they will be automatically redirected to the HTTPS address of your website. After this step, Certbot's configuration is finished, and you will be presented with the final remarks about your new certificate, where to locate the generated files, and how to test your configuration using an external tool that analyzes your certificate's authenticity.

Your certificate is now installed and loaded into Apache's configuration. Try reloading your website using https:// and notice your browser's security indicator. It should point out that your site is properly secured, typically by including a lock icon in the address bar.
In the next and final step, you'll test the auto-renewal feature of Certbot, which guarantees that your certificate will be renewed automatically before the expiration date.

Απαντήστε τις πιο κάτω ερωτήσεις επανάληψης από την ύλη του ΕΠΛ232.

2) Βρείτε τον αριθμό των αρχείων στον τρέχον φάκελο τα οποία έχουν χρόνο τροποποίησης μεταξύ 08.00 – 08.59 π.μ. στις 20 Οκτωβρίου.

3) Βρείτε όλες τις λέξεις που αποτελούνται από ακριβώς πέντε γράμματα από το λεξικό /usr/share/dict/words. Πόσες είναι αυτές οι λέξεις; Εμφανίστε τις ταξινομημένες σε αντίστροφη αλφαβητική σειρά εκτυπώνοντας σελίδα – σελίδα στην οθόνη

4) Βρείτε το συνολικό αριθμό γραμμών των αρχείων του τρέχοντος φακέλου που περιέχουν το string Linux αλλά όχι το string Unix

5) Εκτυπώστε στην οθόνη όλες τις διεργασίες που ανήκουν στον χρήστη root ταξινομώντας τις βάση του χρόνου έναρξης της διεργασίας (ξεκινώντας από την πιο πρόσφατη)

6) Το αρχείο /etc/passwd περιέχει μια γραμμή για κάθε χρήστη του συστήματος. Κάθε χρήστης περιγράφεται από 7 πεδία τα οποία διαχωρίζονται με ‘:’. Τα πεδία αυτά είναι τα Username:Password:UID:GID:UID Info:Home directory:Command/shell. Θέλουμε να δημιουργήσουμε το αρχείο users_info το οποίο θα περιέχει μόνο τα πεδία Username και το UID info σε αλφαβητική σειρά του UID info.

7) Δημιουργήστε το tar αρχείο weekly.tar το οποίο θα περιέχει όλα τα αρχεία (όχι φακέλους και ειδικά αρχεία) του τρέχοντος καταλόγου τα οποία τα οποία έχουν τύχει αλλαγών σε 7 ή λιγότερες μέρες. Χρησιμοποιήστε τα man pages για περισσότερες πληροφορίες για τις επιλογές της δημιουργίας του tar αρχείου.

8) Σβήστε όλα τα αρχεία με όνομα core από το σύστημα.