

Creating an Ubuntu Server 20.04 LTS virtual server in the DEI private cloud. Managing a Linux server from the command line shell. The Apache web server, managing services with the systemctl command. The BASH and command line utilities.

## 1. Using Linux Containers in the DEI private cloud

Virtualization concepts, embracing hardware virtualization (Virtual Machines), operating system virtualization (Containers) and network virtualization (Virtual Switches), are used together to establish what is known as cloud computing.

Cloud computing platforms provide a user friendly way to create and manage virtual servers and virtual networks at a high level without handling on how they are implemented.

Several platforms are publicly available, among others: Amazon Web Services (AWS), Google Cloud Platform, Microsoft Azure, IBM Bluemix, and Alibaba Cloud.

Among others, at least three types of services may be provided:

- IaaS (Infrastructure as a Service)
- PaaS (Platform as a Service)
- SaaS (Software as a Service)

The customer skills required for each service are rather different, for IaaS, usually it's up to customer installing and managing the operating system. For PaaS, the operating system is already installed and is only partially managed by the customer, the customer will be however responsible for installing and managing applications. For SaaS, the customer is provided with an already installed application and is limited to manage that application.

At DEI, an experimental cloud computing platform is available, the front end web page is:

<https://vs-ctl.dei.isep.ipp.pt/>

**Warning:** even though this cloud service is available at public networks (internet), the virtual servers it provides are in a private network environment available within DEI networks only, if you are somewhere on the internet or connected to a wireless network at ISEP, you will need to establish a VPN connection with DEI to access many of the virtual servers features.

### **PRACTICE:**

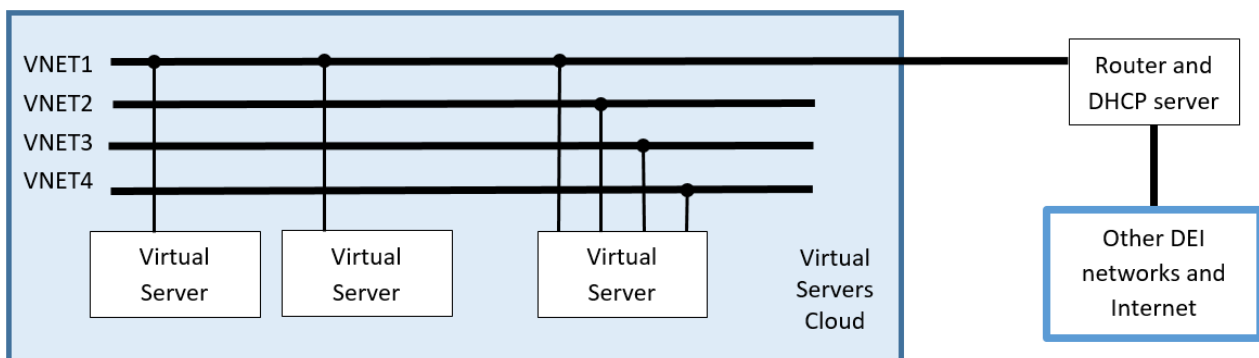
- a) Use your DEI credentials to login, on the start page you have some documentation to read later.
- b) Create your own virtual server:

Go to **Available Virtual Server Templates (VST)**, find template number **13**, the template is named **Apache and others on Ubuntu 20.04 LTS (Focal)** it's a Docker **container**. The type of service provided is Platform as a Service (PaaS).

Click **Create a new VS from this template**.

In this experimental platform, virtual servers may be connected to up to four virtual networks (virtual switches or virtual bridges that are equivalent to physical networks). They are named VNET1, VNET2, VNET3, and VNET4. From those, VNET1 is mandatory because it's through it the virtual servers are accessed for administration purposes.

Of key benefit is every virtual server having a unique static network configuration assigned to it on VNET1, in addition, each virtual server dedicated IPv4 address is mapped in DNS to the virtual server name.



The previously selected template is connected to **VNET1** only, such connection will correspond to a network interface inside the virtual server, and it will be network interface **eth0**.

c) Start and manage the Virtual Server

Click on the virtual server name to manage it, initially the same as the template. Now you are on **Virtual Server Details** page for your virtual server, click the **START** button to boot your Virtual Server, once started new options become available.

**Warning:** the following services are available to you only if you are connected to the DEI private networks infrastructure.

The **root** user (super user) is allowed to access through SSH, a random password was established and is available on the web page, it can be changed. The Virtual Server command line (Shell) may be accessed by using a standard SSH client (e.g. Putty), but it can also be accessed through by using a standard web browser:

Enable the **Terminal in the browser: GoTTY (no login required)**, and then click the link.

You now have a command line (shell) terminal session on your browser, let's do some explorations about the networking environment surrounding your virtual server.

**In the previous lab class, we took the wider perspective of the network infrastructure as a set of interconnected intermediate nodes, it's the network administrator's point of view. Today we take the end node administrator's point of view, we trust the network infrastructure operates appropriately, and we are concerned our end node is configured properly to use such network infrastructure.**

**Intermediate nodes** forward data, for instance, switches and routers are intermediate nodes. **End nodes** are the source and the final destination of the useful data being transferred through the network infrastructure, typical end nodes are servers and workstations.

## 2. Network configurations in Linux

### 2.1. The node's IP addresses

Start by listing your virtual server's **network interfaces**, use the following command:

```
ip addr
```

There's a lot of data to explain here:

- There are 3 network interfaces: **lo**, **docker0**, and **eth0**. However, only **eth0** is connected to the external world (VNET1), the others are internal and used within your virtual server only.
- The **lo** interface stands for loopback interface, the loopback interface exists on most network nodes, it is used for communications within the node itself, it is not connected to anything else.
- The **docker0** interface is a virtual switch interface created by the docker service running inside this virtual server, docker containers running inside this virtual server will be connected to an internal network based on this virtual switch. Even though this virtual server is itself a docker container, it has the ability to run docker containers inside it.
- We can see interfaces have both IPv4 and IPv6 addresses assigned to them, so yes, this is a **dual-stack node**.
- IPv6 addresses started by **fe80::** are self-assigned with link-local scope (link-local scope means they allow communications within the same local network only).
- Only the **eth0** interface is giving access to outside the virtual server. All communications and interactions you are taking with your VS are passing through this **eth0** interface.
- The **eth0** interface has one IPv4 address belonging to network **10.9.0.0/16** and an IPv6 address belonging to the network **fd1e:2bae:c6fd:1009::/64**.

### 2.2. Routing

As we already know, the configuration data listed above allows this node to communicate only with directly connected networks (local networks), in this case they are: **10.9.0.0/16** and **fd1e:2bae:c6fd:1009::/64**.

To reach nodes that don't belong to local networks, neighbour routers must be used, the information about such neighbour routers is stored in routing tables.

Every IP node has routing tables, for an end node like a server they are usually very simple. Use the following two commands to see the IPv4 and the IPv6 routing tables:

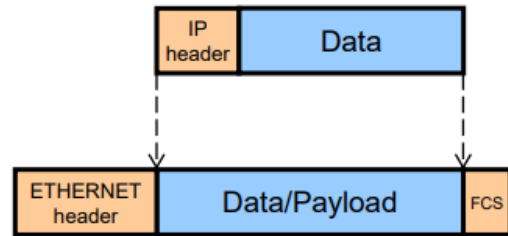
```
ip route
```

```
ip -6 route
```

You may surmise the IPv4 default-gateway is **10.9.0.1** and the IPv6 default-gateway is **fd1e:2bae:c6fd:1009::1**. Notice that the routers in the routing tables are always neighbour nodes, meaning they belong to a local network (a directly connected network).

### 2.3. IP addresses to layer two addresses mapping

Within a local network, IPv4 and IPv6 packets are transported by layer two frames, for such transportation to be possible the layer two destination address has to be properly set in the layer two header so that the packet is delivered where intended.



Two cases may be highlighted:

If the IP packet's destination address belong to the same local network, then the layer two destination address is set to the layer two address of that node, this is because the packet can be directly delivered to the final destination node.

If the IP packet's destination address doesn't belong to the same local network, then the layer two destination address is set to the layer two address of a router that will forward the IP packet.

Either case, given an IP address of a neighbour node that node's layer two address has to be determined, with IPv4 addresses, ARP is used, with IPv6 addresses, ICMPv6 is used. As these mappings between IP addresses and layer two addresses are determined (by using ARP or ICMPv6), they are cached in tables. Use the following command to see such tables:

```
ip neigh
```

Certainly, one node that will be there is the IPv4 default-gateway (10.9.0.1).

Now interact with a new local node:

```
ping -c 2 10.9.20.124
```

Check the neighbour tables again:

```
ip neigh
```

As expected you will see the address targeted by the ping command, this new entry will not persist for ever, if there's no further interaction with that node it will be removed after some time.

### 2.4. DNS node names resolution

As with any network node connected to the internet, your virtual server uses DNS servers to transform node names into IP addresses, let's check a DNS node name we have used in the previous lab class (**rede.dei.isep.ipp.pt**), we can use the **host** command, it's a DNS client:

```
host rede.dei.isep.ipp.pt
```

What have we learned from this command's output:

- The DNS name **rede.dei.isep.ipp.pt** is an alias, not the canonical DNS name of the server node, the canonical (real) name is **vsrv7.dei.isep.ipp.pt**.
- The node **vsrv7.dei.isep.ipp.pt** is dual stack, it has an IPv4 address and an IPv6 address.

Both your virtual server and the **rede.dei.isep.ipp.pt** server are dual-stack, so they can talk with each other either by using IPv4 or IPv6, truly, in this kind of scenario most of the time users don't know whether IPv4 or IPv6 is being used because they refer nodes by DNS names and not IP addresses.

Let's check:

```
ping rede
```

The name **rede**, we already know, stands for the node with DNS name **vsrv7.dei.isep.ipp.pt**, and it's mapped to both an IPv4 and an IPv6 address.

In these cases, **the standard is prioritize IPv6 over IPv4** to allow the IPv6 takeover of the internet.

If you want to test the connectivity through IPv4 you must avoid the DNS resolution of the name and use the IP address:

```
ping 192.168.62.32
```

We can also explicitly force the ping command to use either IPv4 or IPv6:

```
ping -4 rede
```

```
ping -6 rede
```

## 2.5. Setting up a non-persistent network configuration

For the sake of testing, we can change the running network configuration directly from the command line, however these changes will not persist if the server is restarted.

Each virtual server being used by in the class is connected through **eth0** to **VNET1**, so they are all connected to the same physical network.

Over VNET1 there is an already settled IPv4 network: **10.9.0.0/16**. But we can create others.

Let's establish two additional IPv4 networks over **VNET1**, (to which the **eth0** network interface of your server is connected), the networks are going to be **10.200.8.0/24** and **10.200.9.0/24**, some students will use an IPv4 address belonging to one network and others an IPv4 address belonging to the other network:

- If you have an even student number, then you will be using an IPv4 address belonging to network **10.200.8.0/24**.

- If you have an odd student number, then you will be using an IPv4 address belonging to network **10.200.9.0/24**.
- To establish the node number within the network, look again at your student number and extract the two rightmost digits, then remove any leading zero. If you attain zero, use 5 instead, zero is not a valid node.

For instance, student number **9996194** is supposed to use the **10.200.8.94/24** IPv4 address.

**Once you know the IPv4 address you are supposed to use (10.200.X.Y), you may then type the command:**

```
ip addr add 10.200.X.Y/24 dev eth0
```

Let's check the configuration has been enforced:

```
ip addr  
  
ip route
```

**(Please, wait until your classmates have set their own IP configuration)**

Use the ping command to send ICMP echo requests to other IP addresses in your class.

```
ping ...  
ping ...  
...
```

Check that you are able to reach other virtual servers on the same IPv4 network, but not in the other IPv4 network. If your student number is even you are able to reach only even numbers classmates.

**(Please, wait until your classmates are done with the testing)**

Now we are going to make a small change in the configuration: **reduce one single bit in these networks' prefix-length (/23 instead of /24)**.

First, let's remove the address we have added:

```
ip addr del 10.200.X.Y/24 dev eth0
```

And now add the same address again, but with a **/23** prefix-length:

```
ip addr add 10.200.X.Y/23 dev eth0
```

**(Please, wait until your classmates have set the IP configuration)**

Use the ping command to send ICMP echo requests to other IP addresses in the provided list.

```
ping ...  
ping ...  
...
```

Now you should be able to reach every virtual server, simply because now there's only one IPv4 network and not two any longer, the new IPv4 network is: **10.200.8.0/23**

The difference between these network prefixes 10.200.8.0 and 10.200.9.0 lays on 23<sup>rd</sup> bit being zero on the former and one on the later. By setting the prefix-length to 23 bits, that bit no longer belongs to the network prefix, so the network prefix becomes the same.

### 3. HTTP (Hyper Text Transfer Protocol) and the Apache server

HTTP is the protocol through which contents are transferred between **web servers** (HTTP servers) and **web browsers** (HTTP clients) around the internet. HTTPS (HTTP secure) is HTTP running over a secure connection, so in fact it's not a different protocol.

Your virtual server already has an HTTP server running (Apache 2), but we will start by using HTTP clients.

#### 3.1. HTTP clients

Your virtual server (a container) is connected to VNET1 in bridge mode, meaning no address translations (NAT) are being enforced within the DEI networks infrastructure.

If that is so, when we access a server, our real address is not masked and will be visible to the server, let's show this by calling our comrade **rede.dei.isep.ipp.pt**.

First, we will install a **standard web browser** that is able to run in a terminal, it is called **lynx**. In this Linux system, we can manage software packages through the APT (Advanced Packaging Tool).

As with other Linux distributions, for Ubuntu there are several servers in the internet, usually referred to as repositories or simply mirrors, they make available updated software packages. APT stores locally a list of the available software packages in each repository. Before installing software packages it's better to be sure the list is updated:

```
apt update
```

Remember this command doesn't update any software, it updates the list of the available packages and where they are available to be fetched.

Use the **apt** command to install **lynx**:

```
apt install lynx
```

Now use **lynx** to access the web page **https://rede.dei.isep.ipp.pt/myip**

Just type:

```
lynx rede/myip
```

Why is **rede/myip** enough?

We already know the name **rede** is going to be resolved by the DNS system into an IP address.

It's enough because the default DNS domain in your virtual server is **dei.isep.ipp.pt**, and by default lynx adds the **http://** prefix. This web server itself redirects any http request to https, so the browser ends up with **https://rede.dei.isep.ipp.pt/myip**.

We can see your requests, as seen by server **rede.dei.isep.ipp.pt**, seem to be coming from an address that is exactly the IP address your virtual server has. This proves NAT is not being used, also you can see your server is reachable from server **rede.dei.isep.ipp.pt**, your virtual server has replied to the ICMP echo request sent by server **rede.dei.isep.ipp.pt**.

One other useful command to test HTTP servers is **curl**, it's an HTTP client, however, unlike lynx, **curl** is intended mostly for testing and debugging by software developers. It is already installed in your virtual server. Test the same access as before, now with curl:

```
curl rede/myip
```

The curl command also assumes by default the use of HTTP, thus, as before this result in **http://rede.dei.isep.ipp.pt/myip**, however, it doesn't follow the server indication to redirect the requests to HTTPS (this is something standard web browsers are supposed to do).

With curl you must do it yourself:

```
curl https://rede.dei.isep.ipp.pt/myip
```

This is a testing and debugging command, and has many useful features that standard browsers don't have, for instance you can select the use of IPv4 or IPv6:

```
curl -4 https://rede.dei.isep.ipp.pt/myip
```

```
curl -6 https://rede.dei.isep.ipp.pt/myip
```

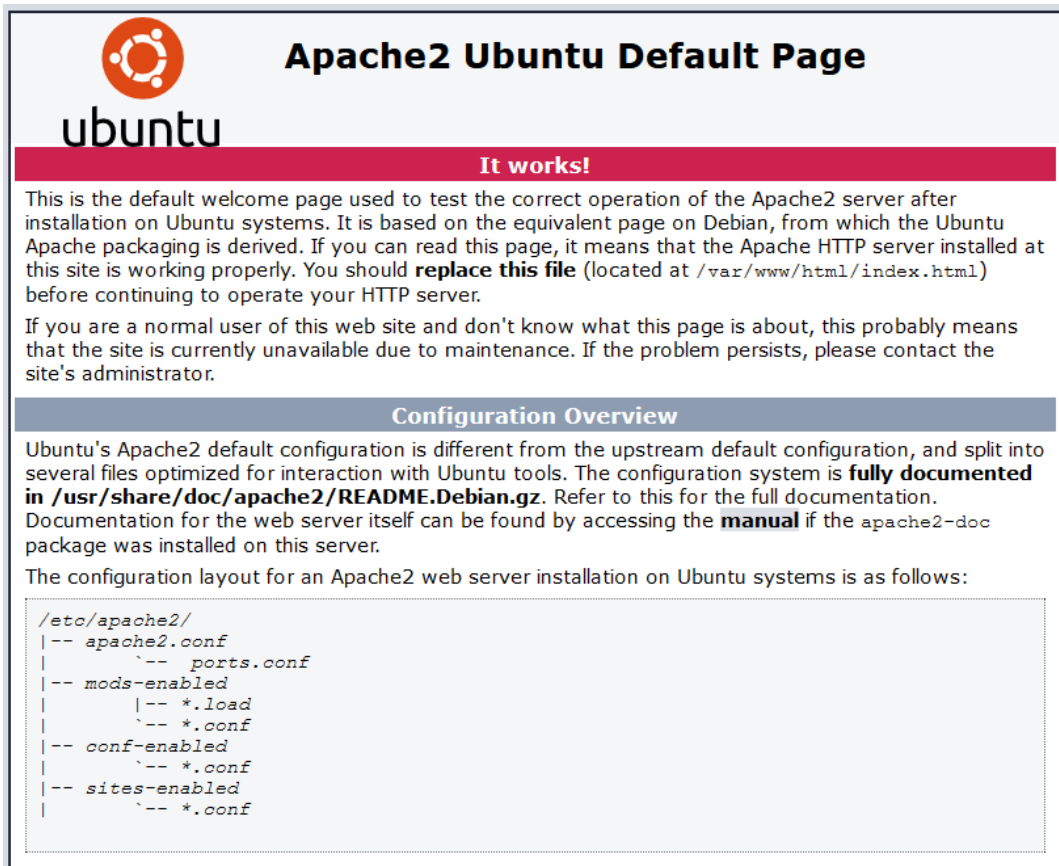
### 3.2. The Apache HTTP server


Your Virtual Server has the Apache web server installed and running, the most basic operation of a web server is providing clients (web browsers) the content of static files.

Those static files are referred by clients following the web server name, if no filename is provided by the client, the server assumes a default filename, usually **index.html**.

Go to your **Virtual Server Details** page and **click the link to access your HTTP server**, you'll get the Apache Default Web Page, for now this is the default page of your web server.





**Apache2 Ubuntu Default Page**

**It works!**

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

**Configuration Overview**

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf
```

The web server searches for filenames referred by clients in a base folder, often called the **document root**, in this specific apache configuration the document root is `/var/www/html`. The `index.html` for the Apache Default Web Page should be there.

Because you are the administrator (root), you can control services running on your virtual server by using the **systemctl** command:

First check the current status of the **apache2** service:

```
systemctl status apache2
```

Now request it to stop running:

```
systemctl stop apache2
```

If you try to reload the Apache Default Web Page you will see it is no longer available. Check the status of the service:

```
systemctl status apache2
```

Now start the service again:

```
systemctl start apache2
```

Everything should now be as before.

When your virtual server boots, the **apache2** service is automatically started, if by some reason you don't want that, then you can disable it:

```
systemctl disable apache2
```

If the **apache2** service is not starting automatically during you server boot, then you can enable it to change that:

```
systemctl enable apache2
```

Let's proceed by creating our own default web page, and meanwhile get familiar with some important concepts and commands.

Many command line operations encompass objects in the filesystem, which as we know is organized as a tree of directories or folders. In MS-Windows systems there may be several filesystems, each starting at one drive letter (e.g. C:\ E:\).

In UNIX systems (that include Linux) a single file system exists, its starting point is the root, represented by a forward slash (/). To absolutely identify an object in the filesystem we can represent its location path starting from the root of the filesystem, this is called the absolute path, such representation is always started by a forward slash (/), for instance **/var/www/html/index.html**.

One concept that exists in command line environments that operate over a file system is the current working directory (CWD), when a user accesses the system in a command line environment the CWD is set to the user's home directory (a private folder belonging to the user).

You may see the CWD by using the **pwd** command, standing for Print Working Directory. You can change the CWD by using the **cd** command, standing for Change Directory.

Check your CWD:

```
pwd
```

List the contents of the **/var/www/html/** folder without changing the CWD:

```
ls -l /var/www/html/
```

**Jump into the apache document root**, this means change the CWD to **/var/www/html/**:

```
cd /var/www/html
```

Check your CWD again:

```
pwd
```

**List the CWD folder's content:**

```
ls -l
```

There it the **index.html** file.

Let's replace it by our own. **First rename it:**

```
mv index.html oldindex.html
```

Now we must use a text editor to create the new **index.html** file, the default text editor in UNIX systems is **vi**, however, it's not easy for beginners, an alternative is **nano**, if required install it:

```
apt install nano
```

Now, create the new **index.html** file:

```
nano index.html
```

Notice that this Virtual Server includes the **Ultra Basic CGI based File Manager**, and it may as well be used to undertake most of these actions.

Establish the following HTML content for the text file:

```
<html><head>
<title>My Web server default page</title>
</head>
<body bgcolor=green>
<h1>My Web server default page</h1>
<hr>
<p>The original Apache Default Web Page is available <a href=oldindex.html>here</a>
<hr>
</body>
</html>
```

Save the file, in **nano** use [CTRL]+[X], then [Y], then [ENTER].

This file uses HTML tags, the main purpose of HTML tags is establishing presentation formatting, so it's mostly about text fonts and colours. The `<a>` tag is used to create a clickable live link, in this case pointing to the old HTML page.

Now, go back to the default page of your web server and **reload it**. Match what you see with the HTML content above, test the link to the old page.

Soon in this course we will be talking about HTML (Hyper Text Mark-up Language) in more detail.

## 4. The BASH and command line utilities

We have already learned about some important commands and concepts in the BASH command line environment, in the next lecture a more systematic approach to some important concepts will be undertaken but meanwhile let's introduce some more commands and concepts.

### 4.1. Users and groups of users

In a multiuser system like Linux, different users with different roles are allowed to interact with the system, because different users have different roles, they also have different permissions (things they are allowed to do).

In a UNIX and Linux systems the user which has all the permissions is the **root** user, often referred as the super user or simply the administrator.

The **id** command tells how you are:

```
id
```

Yes, you are the **root** user, the super user and administrator.

Each user has a unique number in the system, the UID (User Identifier), the **root** user has always UID zero. Each user belongs to a primary group of users, the primary group of the **root** user is group **root**. Despite having the same name, the **root** user and the **root** group are two different things, each group has a unique number in the system, the GID (Group Identifier), and the GID zero is used by the **root** group.

Groups are very useful to assign a set of permissions to several users at a time. Imagine you want the users of a department to have a specific set of permissions over some parts of the system, this is a very common scenario, and the best solution is creating a group, assigning to the group the required set of permissions and making those users members of the group. Now imagine later you have to change that set of permissions, it's easy, you only have to change it in one single place, the group.

Traditionally in Ubuntu systems for each user there's a group with the same name, such group is created along with the user to be the user's primary group.

In this system there's a user named **admin**, check it:

```
id admin
```

This user has the **admin** group as primary group, but it's also a member of the **sudo** group. Being a member of this group turns this user into an administrator because it grants the permission to use the **sudo** command.

The list of users known to the system is stored in the **/etc/passwd** file, it's a text file, each user account is a line with a set of colon separated fields, check its content:

```
cat /etc/passwd
```

The fields are:

- login name
- password hash code
- UID
- GID of the user's primary group
- long name or account description
- user's home directory
- initial program or login shell

Check the lines for user **root** and for user **admin**.

One thing you may have noticed is the **password hash code** is filled with an **x**, for security sake they have been transferred to the shadow file: **/etc/shadow**. Unlike what happens with the **/etc/passwd** file, the **/etc/shadow** file can only be read by the **root** user.

This means you (**root**) can read it, so let's check its content:

```
cat /etc/shadow
```

You will notice only the **root** user and the **admin** user have a password, this means only they can login into your system, nevertheless, as root you can impersonate any user even without a password. Use the `su` command to impersonate the admin user:

```
su -l admin
```

The **-l** option forces the execution of the login scripts to mimic a normal login into the system, check who you are now, check that you can read the `/etc/passwd` file, but not the `/etc/shadow` file:

```
id  
  
cat /etc/passwd  
  
cat /etc/shadow
```

Type **exit** to leave the session as **admin** and return to your **root** session:

```
exit
```

Establish a new password for the **admin** user:

```
passwd admin
```

The list of groups known to the system is stored in the `/etc/group` file, it's a text file, each group account is a line with a set of colon separated fields, check its content:

```
cat /etc/group
```

The fields are:

- group name
- password hash code (currently not used in Linux)
- GID
- comma separated list of users than belong to the group beyond the primary group

Check for instance the **sudo** group.

In Linux, there are several commands available to manage users and groups:

```
useradd; userdel; usermod; groupadd; groupdel; groupmod; adduser; addgroup
```

Create a new user named **scomreduser1**:

```
useradd -m scomreduser1
```

By default a group with the same name as the user is created, it will be the user's primary group.

The **-m** option enforces the creation of the user's home directory, let's check it:

```
ls -l /home
```

Create a new group named **scomredgroup1**:

```
groupadd scomredgroup1
```

Add the **scomreduser1** user to the **scomredgroup1** group, don't change the user's primary group:

```
usermod scomreduser1 -G scomredgroup1
```

Check the created user:

```
id scomreduser1
```

## 4.2. Getting help about a command

Most Linux commands support the **--help** option to provide information about how they operate and possible command line arguments, try getting help about the **ls** command with this tactic:

```
ls --help
```

Because the output may have more lines than the ones being displayed in your window, it may be useful to pass the result through a pipe to the less command:

```
ls --help|less
```

With the less command, among other options, like search a string, you can navigate up and down, use the lowercase **q** to quit and return to the command line prompt.

One other option to get help about a something is searching in in the manual pages, however, in this specific virtual server, the filesystem is minimized to save space and manual pages are not available.

Nevertheless there are online versions of the Linux manual pages, for instance:

<https://man7.org/linux/man-pages/index.html>

## 4.3. The filesystem

We already know in UNIX and Linux there's a single filesystem, the aim of a filesystem is storing objects, for instance files.

One very important type of object is the folder or directory, such object type has the single purpose of containing other objects, which may be themselves folders, thus we end up with a tree of folders. The starting point of this tree is called the **root folder** and represented by the forward slash (/).

When a new filesystem is created it's completely empty, meaning it has no objects, but in fact the root folder is already there.

Objects in the directory tree may be referred by specifying the path starting from the root folder, this is an **absolute path**.

Objects may also be referred by specifying the path starting from the current working directory, this is a **relative path**.

When using a relative path we can specify one level toward the root by using the folder name **../**

**Do on your own:**

- a) List the contents of the **/etc/apache2/** folder using an absolute path (started by **/**).
- b) Print the current working directory.
- c) List the contents of the **/etc/apache2/** folder using a relative path (not started by **/**).
- d) Change the current working directory to **/etc**.
- e) List the contents of the **/etc/pam.d/** folder using a relative path (not started by **/**).
- f) Change the current working directory to your home directory.
- g) Create a sub-folder named **testing123**, use the **mkdir** command.