**Overview**

During the lectures for this module you saw an overview of some key network services – the sort of services that you sit down at a computer to use. However, there are other services that are just as important but are more “behind the scenes” in nature. In this lab we are going to identify those services and get a first-hand look at what they’re doing on a network.

The following lab requires the use of three VMs from the course VM collection and that you open with Oracle VM Virtualbox (your professor will discuss this process with you if needed). Go ahead and download them now: “Network Services Lab - Windows Server.ova”, “Network Services Lab - Windows 10.ova”, and “Network Services Lab - Linux (Fedora 36).ova”.

Once again, I have highlighted the spots where you need to provide answers in red.

**Outcomes**

This lab is designed to familiarize you with some of the basic network services you can run and/or access on a modern OS. You will also do some personal research to answer additional lab questions related to the exercises.

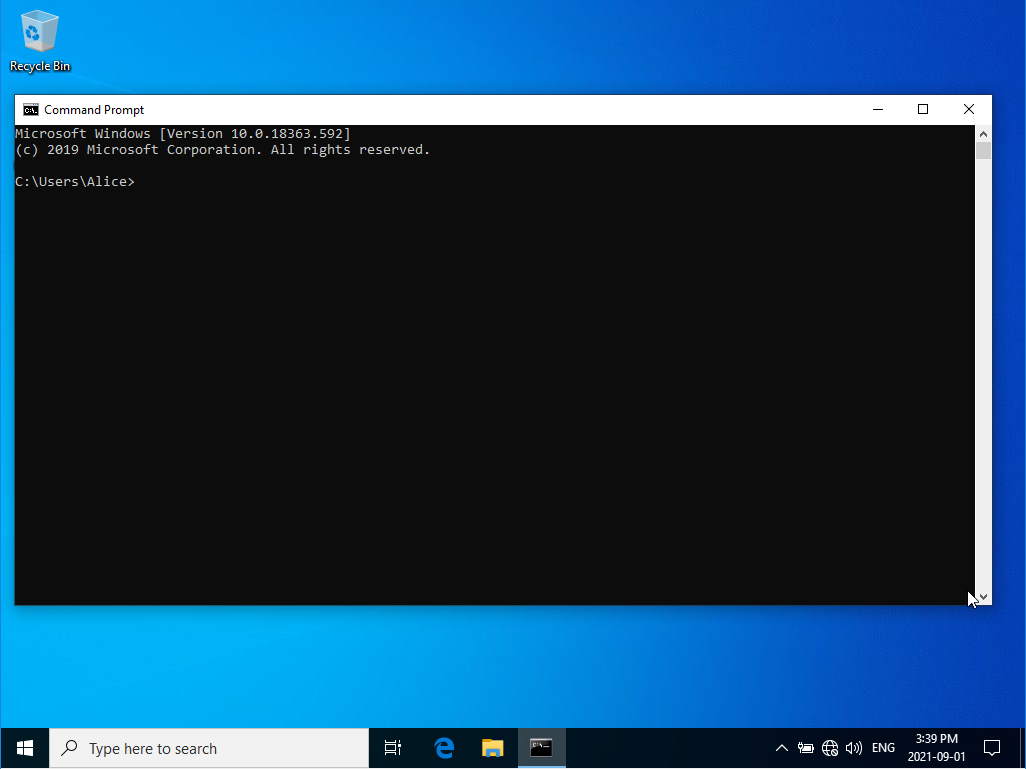
|  |  |
| --- | --- |
| Elements of Performance | Related Course Learning Outcomes |
| EOP 4.4. Learn about directory services and other foundational OS/network services | CLO 4. Identify major network services |

**Time on Task**

|  |  |
| --- | --- |
| Learning Activity | Approximate Time |
| Perform hands-on lab tasks and answer lab questions | 30 min |
| Read, research, and write answers to research section | 15 min |

**Lab Instructions**

1. Ensure that you have successfully downloaded the three VM appliance files that we’re going to use today:  
     
   Network Services Lab - Windows Server.ova  
   Network Services Lab - Windows 10.ova  
   Network Services Lab - Linux (Fedora 36).ova
2. Double click the Windows 10 ova and import it into VirtualBox (or import it via the menu).
3. Start the Windows 10 VM and log in as user **Alice** with the password **userpass**  
   (yes, this is a terrible password that you should never use in the real world, but it makes things simple for our labs).
4. When the VM desktop appears, open a command window by clicking the Windows logo (aka start button) at the bottom left of the screen and typing **cmd** then clicking on the “Command Prompt” item that appears. When started your VM Desktop will look something like this:



1. Now, let’s check to see what IP address has been assigned to this node. Remember, all network nodes need a working IP address to be able to send and receive data on the network.  
     
   Type the command: **ipconfig**  
   You should see a response ***similar*** to the following:  
     
   Windows IP Configuration  
     
   Ethernet adapter Ethernet:  
     
    Connection-specific DNS Suffix . :  
    Link-local IPv6 Address . . . . . : fe80::fc12:55f:1c52:7bfe%4  
    Autoconfiguration IPv4 address. . : 169.254.123.254  
    Subnet Mask . . . . . . . . . . . : 255.255.0.0  
    Default Gateway . . . . . . . . . :

If your output is very different, or you get an error, please ask your Professor for guidance.

Let’s talk about those network addresses for a moment.   
  
The IPv4 address looks good, and is a proper address, but we do have a bit of a problem here. First, as noted in the output of our ipconfig command, the address was assigned via *autoconfiguration* by the Windows 10 OS itself, rather than being a value assigned or reserved by a network administrator – this is risky, because another node could end up with the same value (it probably won’t, but *probably* is a dangerous word in our industry…”Your personal data is probably safe” is not acceptable these days). The other issue is that this address is in the 169.254.x.x block, which is known as an *APIPA (Automatic Private IP address)* address. Likewise, the IPv6 address is stated to be a *link-local* address, and is in the fe80::/10 block (this means the same thing in IPv6 lingo).  
  
These ranges are not routable – that is, routers do not forward packets coming from one of these address blocks because the addresses are not guaranteed to be unique beyond their network segment. These are a type of default address that the OS applies when it doesn’t have a valid address assigned to a network interface and no address is passed to it by the network service that issues them…  
  
Aha! Now we see why we’re here.

1. Import and start the Windows Server ova. There is no need to log in – just have it up-and-running. We’ll wait…  
     
   *Hmmm hmmmmmm hmmmm. Doodoodoo doooooooooo hmm hmm.  
     
   Say, did you hear the one about the tenant who never quite finished their E-mail to their landlord about a broken window? It was saved in their drafts folder…*  
     
   When it’s running, proceed to the next step.
2. After the Windows Server has been running for a minute or two, go back to your Windows 10 VM.  
     
   Re-run the **ipconfig** command. You may notice a few changes!  
   Your output will look similar to:  
     
   Windows IP Configuration

Ethernet adapter Ethernet:

Connection-specific DNS Suffix . :

Link-local IPv6 Address . . . . . : fe80::fc12:55f:1c52:7bfe%4

IPv4 Address. . . . . . . . . . . : 192.168.0.11

Subnet Mask . . . . . . . . . . . : 255.255.255.0

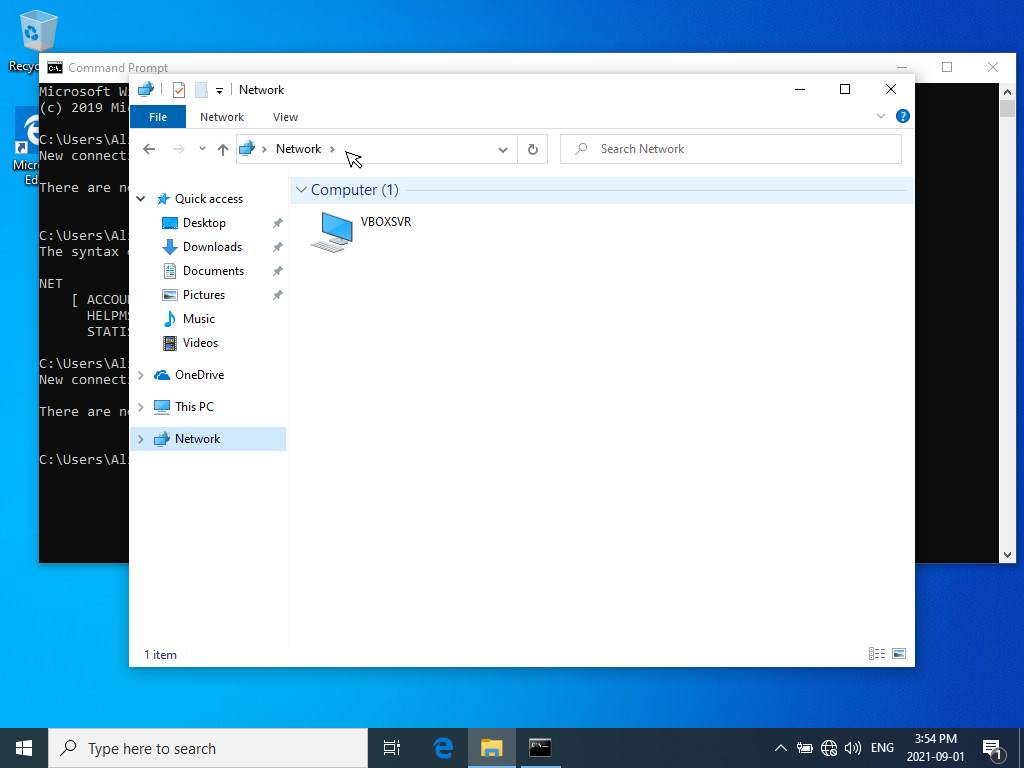
Default Gateway . . . . . . . . . : 192.168.0.1

You can now see that the IPv4 address is now in a different range, that the subnet mask is different, and that the default gateway has been set.  
  
So, where did this change come from? It’s no coincidence that it happened after the Windows Server was started…  
  
The Windows 10 machine is configured to request its IP address from a DHCP server. When you first started that VM there was no DHCP server on its network that could answer the request, so Windows 10 defaulted to using the APIPA range address. Then you started the Windows Server – which has DHCP services running! Because the Windows 10 VM is periodically checking for a DHCP response it sees the reply from the server and configures itself with the IP address it was assigned. Magic!

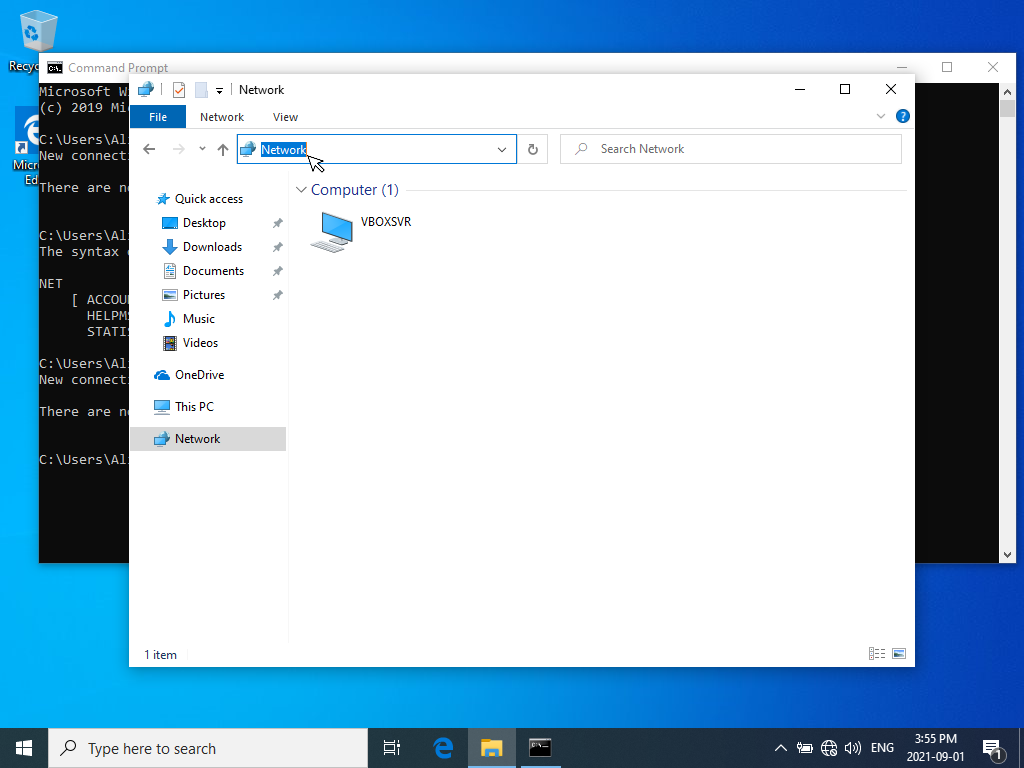
DHCP services is only one of an array of important network services but you can’t get very far on a network without a proper IP address, so it’s pretty fundamental. Let’s see if any other network services are available to us.

1. Open the File Explorer on the Windows 10 VM (use the icon on the task bar, or go through the Start menu as we did before).  
     
   Once it’s open, click on the “Network” item in the “Quick Access” area on the left-hand side. The workstation will then scan the network for open shares. It will likely find one called “VBOXSVR” that’s being provided by the VirtualBox environment. We don’t care about that one today.  
     
   Instead, let’s see if the Windows Server is providing any interesting file shares!

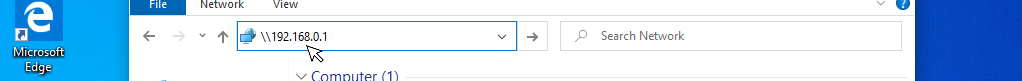
Click on the address bar near the top of the File Explorer window (it’s just under the menu bar):



Once you click there, the word Network will be highlighted:

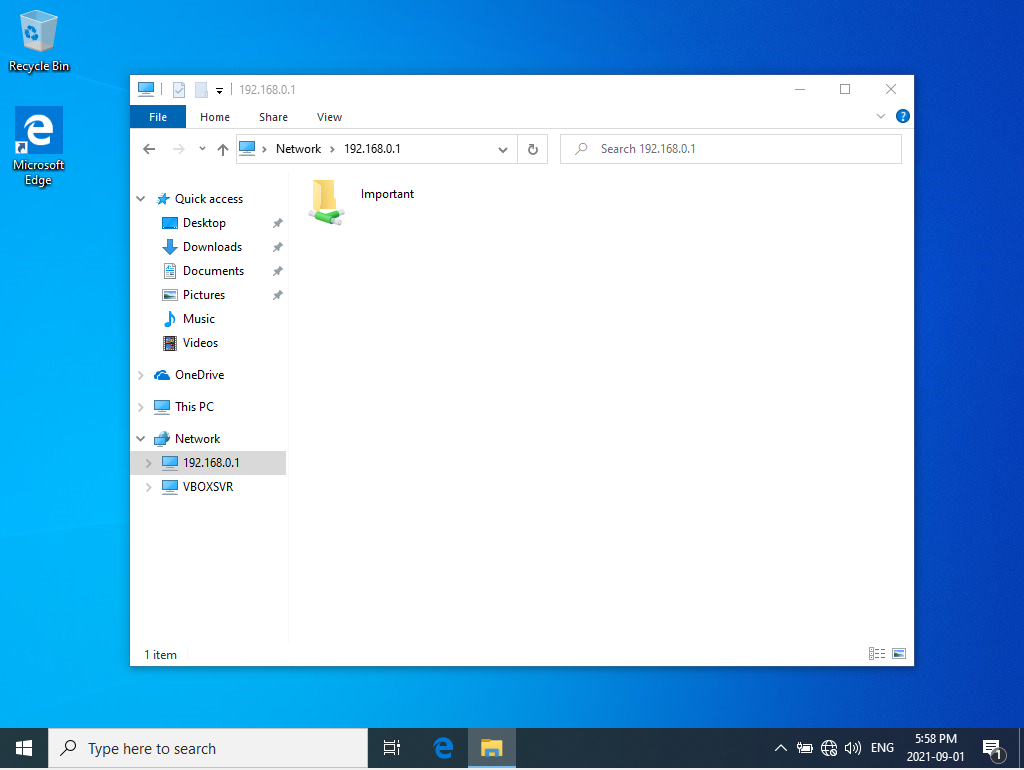


Replace the word Network by typing **\\192.168.0.1** (that’s the IP address of the Windows Server, preceded by two \ characters which tells File Explorer that we’re looking to find a file share):



Press Enter.

You should now see a network file share that is clearly quite **Important**:



The icon here is trying to show you that it is a folder, similar to one local to the machine itself, but on the network (those green and white pipes are supposed to be network cabling – it sort of makes you think that Microsoft has never seen a network cable).

Double-click on the folder. You are now *browsing* the contents of a folder that is not on this VM – it’s actually on the Windows Server VM. Feel free to open the file that’s in there by double-clicking on it. Evidently it’s pretty important…

1. Pause the Windows Server by clicking on the “x” in the upper-right corner of the window, then choose “Save the machine state” from the list, then click “OK”.  
     
   The Windows server has just gone to sleep – it’s a lot like what happens when you close the lid on your laptop.
2. Return to the Windows 10 VM and try to open the file in the network folder.

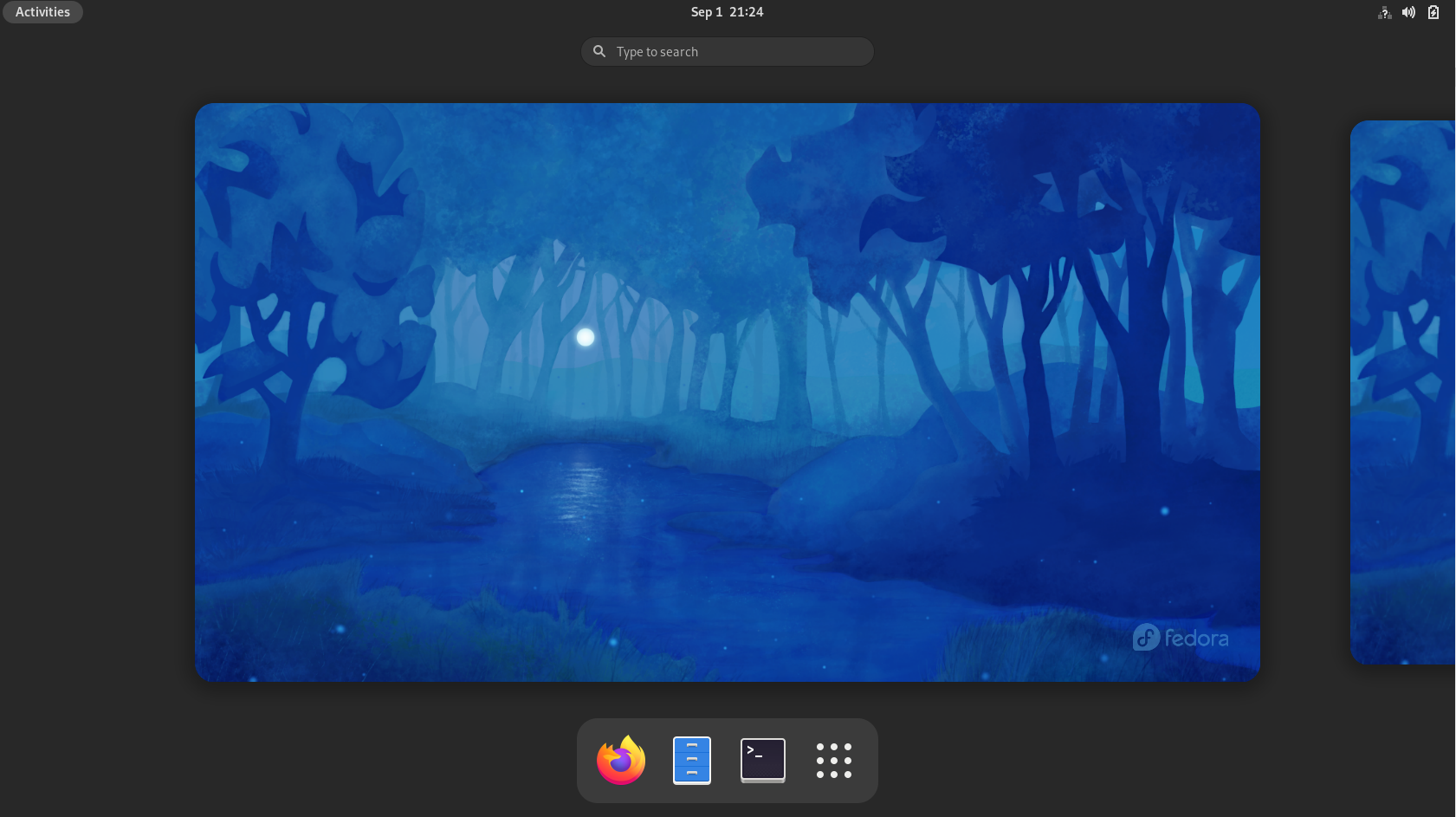
You’ll get slightly different results if you opened the file before vs. someone who has never opened the file, but you should see a message saying “We can’t open this file.”  
  
This is not a surprise, but it’s good that we checked. Close the application with the error message in it.

1. Go to the Oracle VM VirtualBox Manager application window and locate the **Network Services Lab - Windows Server** VM. It should have a state of “Saved”. Click on it and then click the “Start” button.  
     
   You should see a dialog box reporting that it is “Restoring Virtual Machine”, and you will quickly find the VM back up and running.
2. Return to the Windows 10 VM and try to access the file again.  
   This attempt should be more successful!  
     
   Close the Photos app (or if you’re enjoying it, leave it open for a while if you’ve got the spare CPU and memory). When you’re ready, shut down the Windows 10 VM the way you would shut down any Windows 10 system.
3. Import and start the Linux ova. Again, we’ll wait…

*Doo Doo Doo, doo doo doo. Hmmmmmmmm hmm hmm hmm.  
  
Say, did you hear the one about the Linux system administrator who became more confident in public? They really came out of their shell…*

When it’s up, login as user **Alice** by clicking on that name on the login screen. Use the password **userpass** (yes, I know).

You should see the following appear (this is called the Gnome desktop – there are several desktop models in Linux, and this is the one that the Fedora distribution has chosen as their default):



If you ever need to return to this screen (it’s called the “activities view”) just click the “Activities” button at the upper-left corner of the desktop.

Click the “Terminal” icon (it’s the black box in the bottom panel to the right of the blue filing cabinet). The activities view will change to the regular desktop view, but you should have a terminal window open. The terminal is similar to the windows command prompt.

1. Let’s make sure that we have a working network address. Type the command **ifconfig** (yes, it’s almost the same command as you used in Windows, but a little different – you need to make sure you type it correctly).  
     
   You will get more output than you did from the Windows version of the command, and you will see data about a second network interface that you can ignore. The output will look something like this:

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 192.168.0.10 netmask 255.255.255.0 broadcast 192.168.0.255

inet6 fe80::ea86:b669:257f:9017 prefixlen 64 scopeid 0x20<link>

ether 08:00:27:16:ae:c8 txqueuelen 1000 (Ethernet

RX packets 38 bytes 5651 (5.5 KiB)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 117 bytes 13271 (12.9 KiB

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 6553

inet 127.0.0.1 netmask 255.0.0.0

inet6 ::1 prefixlen 128 scopeid 0x10<host>

loop txqueuelen 1000 (Local Loopback)

RX packets 28 bytes 2912 (2.8 KiB)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 28 bytes 2912 (2.8 KiB)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

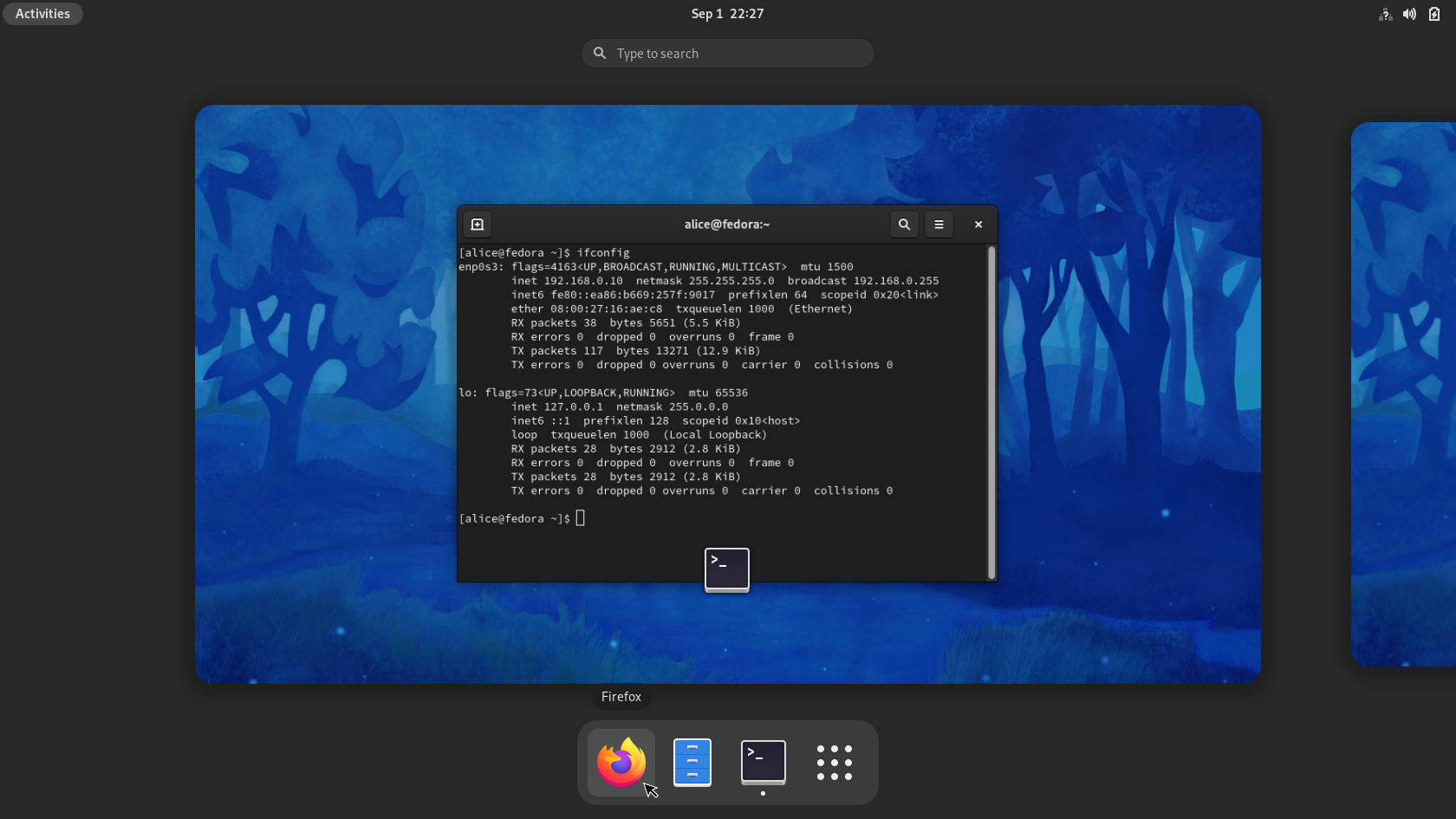
We’ll know that things are working fine if the first interface (enp0s3 is  
the one in this output – you may have a slightly different value) shows  
“inet 192.168.0.*something* netmask 255.255.255.0 broadcast 192.168.0.255”

If you get error messages or something wildly different you should ask your Professor…

This is actually pretty interesting – the Windows Server VM’s DHCP network service is issuing the Linux VM its IP address configuration. You may be surprised to hear that Windows can work with Linux like that, but any device running a proper DHCP service can give addresses to any DHCP client on any kind of device. In fact, if you have an Internet connection at home the router (whether it’s a separate device or something integrated into your DSL or cable modem) is running DHCP services to give your devices (e.g. your phone, your computer, etc.) their IP address settings. Oh, and by the way, that router is likely running a Linux-based OS…

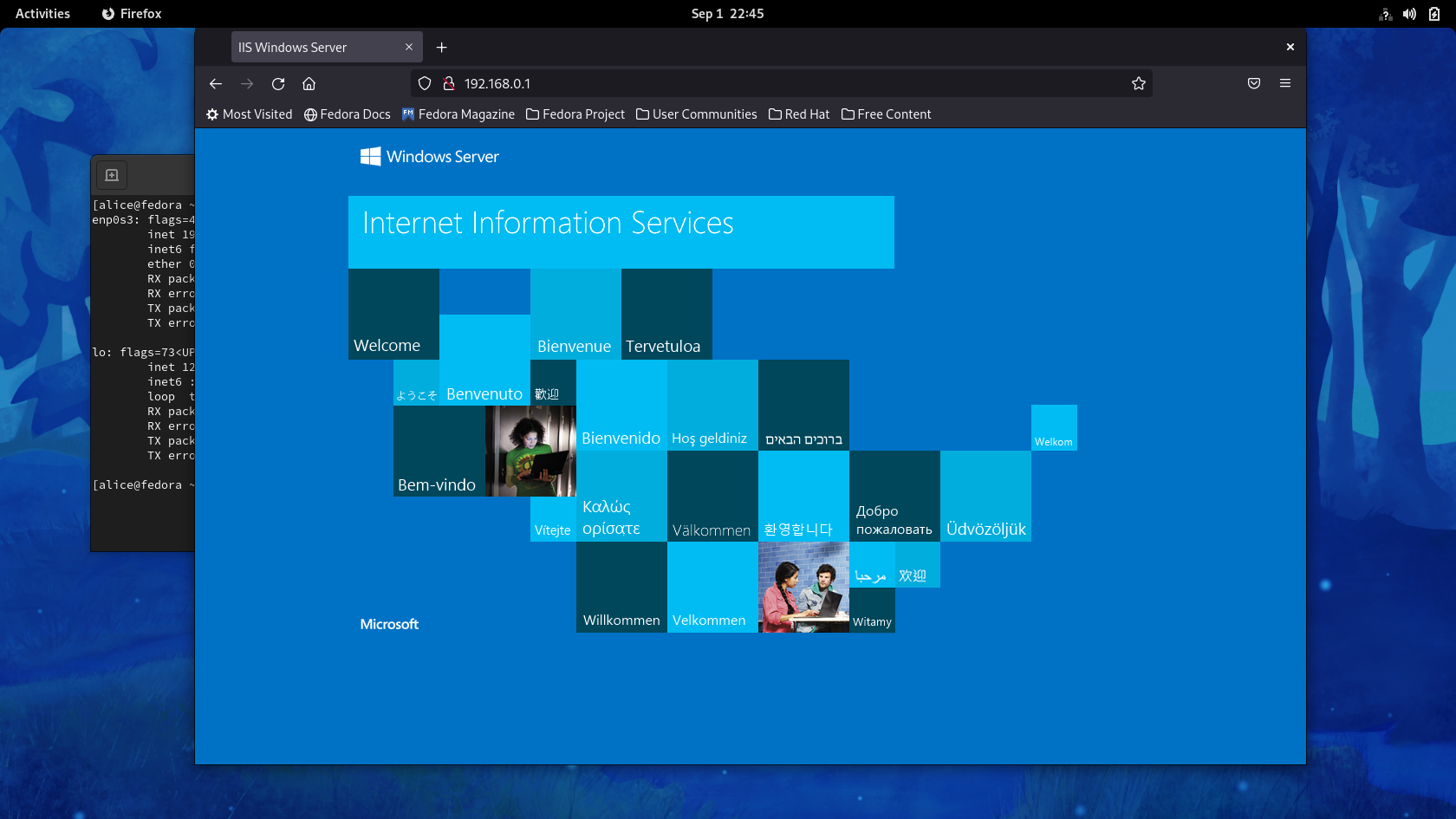
1. Now, let’s see if there’s another network service we can access!

Click on the Activities button to bring up the activities view, then click the Firefox icon (the swirly orange thing):



The activities view will vanish and you’ll quickly see the Firefox browser appear. It may be a more-or-less blank window, or you may have some cached data from when the VM was last configured.  
  
Enter the URL **192.168.0.1** (yes, the IP address of the Windows Server).

You should now see the default page for the web server service running on the Windows Server VM:



1. Shut down the remaining VMs. The Windows Server can be shut down using the method we used in todays lab or you can bring up the login screen by going to the “Input” menu, selecting “Keyboard”, then “Insert Ctrl-Alt-Del”, then clicking on the icon at the bottom-right of the screen (it’s a circle with the vertical line intersecting it) and selecting “Shutdown”, then clicking the “Continue” button. The other OSes can be shut down from their desktop interfaces or by clicking the “x” at the upper-right corner of the VM window, picking “Send the shutdown signal” from the list, then clicking the “OK” button (if this does not work on the Windows 10 VM, the same process you used for the Windows Server VM will work).

**Lab Questions**

1. What *network services* did we investigate today?  
     
    DHCP: assigning Ips, subnets and gateways.  
   SMB/CIFS: for file sharing  
   HTTP(Web Server): accessed through the browser
2. Are these the only important network services? If so, why? If not, what are some of the other ones?  
     
    No. Other important network services include DNS for resolving domains. NTP, which syncs clocks. SMTP for sending mail. FTP for robust file sharing. SSH for securely connecting to remote devices.
3. Can a network service running on one type of OS be accessed and used by a completely different OS? If so, what evidence did we see of that in our lab today? If not, how would we go about provide OS-specific services?  
     
    Yes! That is a fundamental property of network services to be interoperable across different VMs.  
   DHCP worked on windows and Linux, SMB would behave the same.  
   HTTP is accessed by almost every OS regardless of host.  
   As long as OS follow the same rules they can be interoperable.
4. We used Fedora Linux in this lab. Who makes it? What is its purpose?  
     
    Fedora is maintained by a community of Linux devs. RedHat Enterprises is a main sponsor. It’s mainly for accessible and a free OS for hardware, clouds, and containers. Because of its opensoure nature and diverse community it is well maintained and feature rich.
5. If you did this lab away from the college, please let me know how well the VMs worked for you on your computer. I’d also like to know about your computer – ideally I’d like to know the brand, OS (Windows, MacOS, etc.), CPU model (if you know it), and the amount of RAM.  
     
   I ran this lab from the school PC. These were the specs:

Device brand Dell

Processor Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz 2.90 GHz

Installed RAM 16.0 GB (15.7 GB usable)

System type 64-bit operating system, x64-based processor

Edition Windows 10 Education

Version 22H2

Congratulations! You’ve successfully made it to the end of this lab! Yay!

Now, save this file and upload it to the submission link on the course page. If you’re not sure how to do that, please ask! That’s what the Professor is for!