12.5.1 Hostname and DNS Configuration

Click one of the buttons to take you to that part of the video.

Hostname and DNS Configuration 0:00-1:01

When you're configuring IP settings on a Linux network interface, there are two key parameters that you need to remember to set. First of all, you need to specify a hostname for the system, and you also need to specify the IP address of the DNS server you want to use for name resolution. If you've ever used a web browser before, and who hasn't, you know that you can use domain names to navigate to a particular host on the web. For example, if you wanted to visit the Amazon website, you'd go to www.amazon.com. But be aware that your Linux system actually can't work with these alphanumeric domain names. We like them, we're used to them, but the system has no idea what you're talking about. When you open up a web browser and you enter in a URL in the URL field, your system has no clue where to go to get the requested information. In order for this to work, your local system has to first resolve that alphanumeric domain name into a binary IP address that it can use.

/etc/hosts 1:02-4:00

There are, actually, two different ways that this name resolution can occur on a Linux system. In the old days, basic hostname to IP address mapping was performed in this file right here: /etc/hosts. The contents of the host file is really pretty basic. It says, "This IP address maps to this name, as well as this one right here." We'll talk about what that means in just a second. I want you to be aware before we go on, however, that the /etc/hosts file is still used on Linux systems, even though this is the oldest mechanism around for name resolution. In fact, the /etc/hosts file is the first name resolver that is used by default on Linux systems-- and on most other operating systems, for that matter. When you try to access a URL on the Internet, the first place that Linux is going to look is in this old file right here. And only if a record for the requested domain name doesn't exist in this file will the operating system then try to resolve that hostname, using a DNS server.

Why do I bring this up? It's because you need to manage this host file very carefully. And frankly, most system administrators-- myself included--forget that it's even there, because we really don't use it very much. The problem here is that many network hacks can exploit this functionality of the operating system. Think about it. All you would have to do is write bad information to this hosts file, maybe by visiting a malicious website, or maybe by installing some application that has malware embedded in it. That website or that malware could try to rewrite your hosts file with names mapping to commonly used websites, like Amazon or eBay, and actually point you to fake websites on the Internet that look like your favorite site, but instead are elaborate phishing websites that are designed to steal your personal information. Monitor this file carefully and make sure that it doesn't contain malicious mappings.

With that warning in mind, let's take a look at the syntax that you use within the hosts file. It's pretty simple. As we said a minute ago, you enter the IP address of the host first, you enter the hostname, and then, optionally, you can enter an alias. In this case, we're saying that the hostname fs5.corpnet.com maps to an IP address of 192.168.1.1. This string of text your system really can't use for sending data across the network. This IP address, however, it can use. It can figure out how to get packets from your system to that destination IP address. Over here, we have an optional alias that you can use. In this case, we're just saying that you can either type 'fs5.corpnet.com' and have it map to 192.168.1.1, or you can just type 'fs5' and it will also map to 192.168.1.1.

DNS Name Resolution 4:01-9:26

Back in the early days of IP networking, it was the hosts file that we used to resolve hostnames into IP addresses. However, today it really isn't feasible to use the hosts file as the sole means of name resolution. Imagine how big that file would have to be to resolve all the domain names used by all the hosts on the Internet. In addition, those hostnames on the Internet are always changing--IP addresses are changing, hostnames are changing. Can you imagine how much work it would take to manually add, remove, and modify hostname mappings whenever somebody, somewhere changed an IP address or a hostname on the Internet? It would be an absolute nightmare. It would be bad enough trying to keep you own host file up to date. Imagine if you were responsible for managing a thousand systems, and each had individual hosts files that had to be kept up to date. A better option, and the one that we use today, is to submit the domain name that you need resolved into an IP address to a DNS server. When a DNS server receives a name resolution request, it matches the hostname and the domain name submitted with the IP address that maps to it, and returns it back to the requesting system. Then your system can contact the specified host address, using the IP address instead of the domain name.

Here's how it works. We have our workstation A, down here, and it needs to resolve a hostname for some URL. Say we're trying to reach www.corpnet.com. Well, the system needing to resolve the hostname will send that request to the DNS server that it has been preconfigured to use. And it does so over IP port 53. At this point, the DNS server has to make a decision. If the DNS server is authoritative for the zone where the requested hostname resides, it will respond with the appropriate IP address. When I say authoritative, a DNS server is considered to be authoritative if it contains a record for the hostname being requested in its own internal database of name mappings. In this example, we're trying to resolve www.corpnet.com into the appropriate IP address, but this DNS server only has records for the eastsim.com domain

and the westsim.com domain. It actually doesn't know where to get information about www.corpnet.com. It looks in its own database and says, "I don't know. I don't have any records for corpnet.com." If it did, it would just go ahead and respond to the workstation. But because it doesn't, it's got to go find out who does have that information.

In this situation, the DNS server is going to send a request to a root-level DNS server out on the Internet. There are 13 of these root-level DNS servers out on the Internet, and every DNS server in the world is preconfigured, automatically, with the IP addresses of these root-level DNS servers. These root-level DNS servers are configured with records that resolve to authoritative DNS servers for each top-level DNS domain, such as .com, .gov, .edu, and so on. This root-level DNS server responds back to your DNS server with the IP address of a DNS server that is authoritative for just the top-level domain of the domain name that you're trying to resolve. In this situation, the top-level domain is .com. The root-level DNS server is going to return an IP address of a DNS server that is authoritative for the .com domain. Then your DNS servers send a name resolution query to the DNS server over here that is authoritative for the hostname's domain--in this case, .com. More than likely, this DNS server doesn't have a mapping for www.corpnet.com. All it knows is where the authoritative server for the corpnet.com domain is. It will respond back with yet another IP address to your DNS server that says, "Hey, for information about corpnet.com, here's an IP address that you can go to." Then your DNS server contacts that authoritative DNS server. This server is authoritative; that means it does have a record for the hostname that you're trying to resolve. In this scenario, it does have a record for www.corpnet.com, so now it responds back to your DNS server with the appropriate IP address for this hostname. Finally, then your DNS server returns that IP address back down to your workstation, and then your workstation can contact www.corpnet.com directly, using its IP address. Something you might want to remember here is the fact that after your DNS server has gone through this entire process-- trying to resolve www.corpnet.com into an IP address-- it's actually going to cache that IP address locally, right here. That way, if your system-- or any other system, for that matter-- wants to resolve that hostname again into an IP address, it doesn't have to go through this whole convoluted process. It just pulls it out of its cache and says, "Oh, I already know who that is. Here's the IP address." It saves a little bit of time and saves network bandwidth.

/etc/resolv.conf 9:27-11:10

In order to make this entire system work, you have to provide your Linux system with the IP address of the DNS server that it can send name resolution requests to. And this is configured in this file right here: /etc/resolv.conf. Notice that there is no E right here. That is a common mistake. It is not "resolve" (with an E) .conf, it's "resolv" (with no E) .conf. This file defines the search prefixes, as well as the name servers that the Linux system is going to use. Here is some sample content. This parameter right here, Search, specifies the domain name that should be used to automatically complete incomplete hostname requests. For example, let's suppose I wanted to resolve fs5. All I did was say, "Hey, I need to get data to a host with the name of fs5." Well, because I didn't provide a domain name with the hostname, this parameter right here will automatically populate it with this domain name. The request would go to the DNS server as not "fs5" but "fs5.corpnet.com." And, of course, if I had specified the full domain name, then this parameter would be ignored, and we would just submit the full domain name to the DNS server. The next two parameters, the name server parameters, are used to specify the IP addresses of the DNS servers that you want to use for name resolution. And you can configure, actually, up to three different DNS servers. This is a good idea, because if this first server here fails or doesn't resolve or is otherwise unreachable for some reason, it will just go to the next one and it will try it, and it will keep going down the list until one of them responds. The syntax, as you can see, is pretty simple. We just enter 'nameserver', and then the IP address of the name server.

/etc/nsswitch.conf 11:11-12:07

You can use this file right here, the /etc/nsswitch.conf file, to determine the order in which services will be used for name resolution. This nsswitch stands for name service switch. There are two specific lines that you want to look at-- hosts and networks. These two entries specify the order in which services will be used to resolve hostnames into IP addresses. Notice that by default, it says files, and then it says DNS for both hosts and networks. What this means is that the /etc/hosts file will be searched first. Only if a name mapping is not found in the hosts file will the request be sent to the DNS server. You can change this, if you want, by switching the order around; in which case, we would send the request first to the DNS server, and only if the DNS server is unable to respond would we use the hosts file.

Summary 12:08-12:12

That's it for this lesson. In this lesson we reviewed name resolution using first the hosts file and then also using DNS servers.
