Internet Protocol Troubleshooting Commands

Objectives

After completing this lab, you should be able to:

- Practice troubleshooting commands
- Identify how you can use these commands in customer scenarios

Duration

This lab requires approximately **30 minutes** to complete.

Scenario

You are a new network administrator who is troubleshooting customer issues.

AWS service restrictions

In this lab environment, access to AWS services and service actions might be restricted to the ones that you need to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that this lab describes.

Accessing the AWS Management Console

1. At the top of these instructions, choose Start Lab to launch your lab.
   A **Start Lab** panel opens, and it displays the lab status.

   > **Tip**: If you need more time to complete the lab, choose the Start Lab button again to restart the timer for the environment.

2. Wait until you see the message *Lab status: ready*, then close the **Start Lab** panel by choosing the **X**.

3. At the top of these instructions, choose AWS .
   This opens the AWS Management Console in a new browser tab. The system will automatically log you in.

   > **Tip**: If a new browser tab does not open, a banner or icon is usually at the top of your browser with a message that your browser is preventing the site from opening pop-up windows. Choose the banner or icon and then choose **Allow pop ups**.

4. Arrange the AWS Management Console tab so that it displays along side these instructions. Ideally, you will be able to see both browser tabs at the same time so that you can follow the lab steps more easily.

Task 1: Use SSH to connect to an Amazon Linux EC2 instance

In this task, you will connect to a Amazon Linux EC2 instance. You will use an SSH utility to perform all of these operations. The following instructions vary slightly depending on whether you are using Windows or Mac/Linux.

Windows Users: Using SSH to Connect

💬 These instructions are specifically for Windows users. If you are using macOS or Linux, [skip to the next section](#).

5. Select the Details drop-down menu above these instructions you are currently reading, and then select Show . A Credentials window will be presented.

6. Select the **Download PPK** button and save the **labsuser.ppk** file.

   Typically your browser will save it to the Downloads directory.

7. Make a note of the **PublicIP** address.

8. Then exit the Details panel by selecting the **X**.

9. Download **PuTTY** to SSH into the Amazon EC2 instance. If you do not have PuTTY installed on your computer, download it here.

10. Open **putty.exe**

11. Configure PuTTY timeout to keep the PuTTY session open for a longer period of time.:

    - Select **Connection**
    - Set **Seconds between keepalives** to `30`

12. Configure your PuTTY session:

    - Select **Session**
    - **Host Name (or IP address):** Paste the **Public DNS or IPv4 address** of the instance you made a note of earlier.
      Alternatively, return to the EC2 Console and select **Instances**. Check the box next to the instance you want to connect to and in the *Description* tab copy the **IPv4 Public IP** value.
    - Back in PuTTY, in the **Connection** list, expand ⊞ **SSH**
    - Select **Auth** *(don't expand it)*
    - Select **Browse**
    - Browse to and select the lab#.ppk file that you downloaded
    - Select **Open** to select it
    - Select **Open** again.

13. Select **Yes**, to trust and connect to the host.

14. When prompted **login as**, enter: `ec2-user`
    This will connect you to the EC2 instance.

15. Windows Users: Select here to skip ahead to the next task.

macOS 🍎 and Linux 🐧 Users

These instructions are specifically for Mac/Linux users. If you are a Windows user, skip ahead to the next task.

16. Select the ⬛ Details ⬛ drop-down menu above these instructions you are currently reading, and then select ⬛ Show ⬛. A Credentials window will be presented.

17. Select the **Download PEM** button and save the **labsuser.pem** file.

18. Make a note of the **PublicIP** address.

19. Then exit the Details panel by selecting the **X**.

20. Open a terminal window, and change directory `cd` to the directory where the *labsuser.pem* file was downloaded. For example, if the *labuser.pem* file was saved to your Downloads directory, run this command:

    ```
    cd ~/Downloads
    ```

21. Change the permissions on the key to be read-only, by running this command:

    ```
    chmod 400 labsuser.pem
    ```

22. Run the below command *(replace **\<public-ip\>** with the **PublicIP** address you copied earlier)*.
    Alternatively, return to the EC2 Console and select **Instances**. Check the box next to the instance you want to connect to and in the *Description* tab copy the **IPv4 Public IP** value.:

```
ssh -i labsuser.pem ec2-user@<public-ip>
```

23. Type `yes` when prompted to allow the first connection to this remote SSH server.
    Because you are using a key pair for authentication, you will not be prompted for a password.

Task 2: Practice troubleshooting commands

Recall

Some layers have commands related to them to help with troubleshooting. The following is an example of how the troubleshooting commands flow with the Open Systems Interconnection (OSI) model:
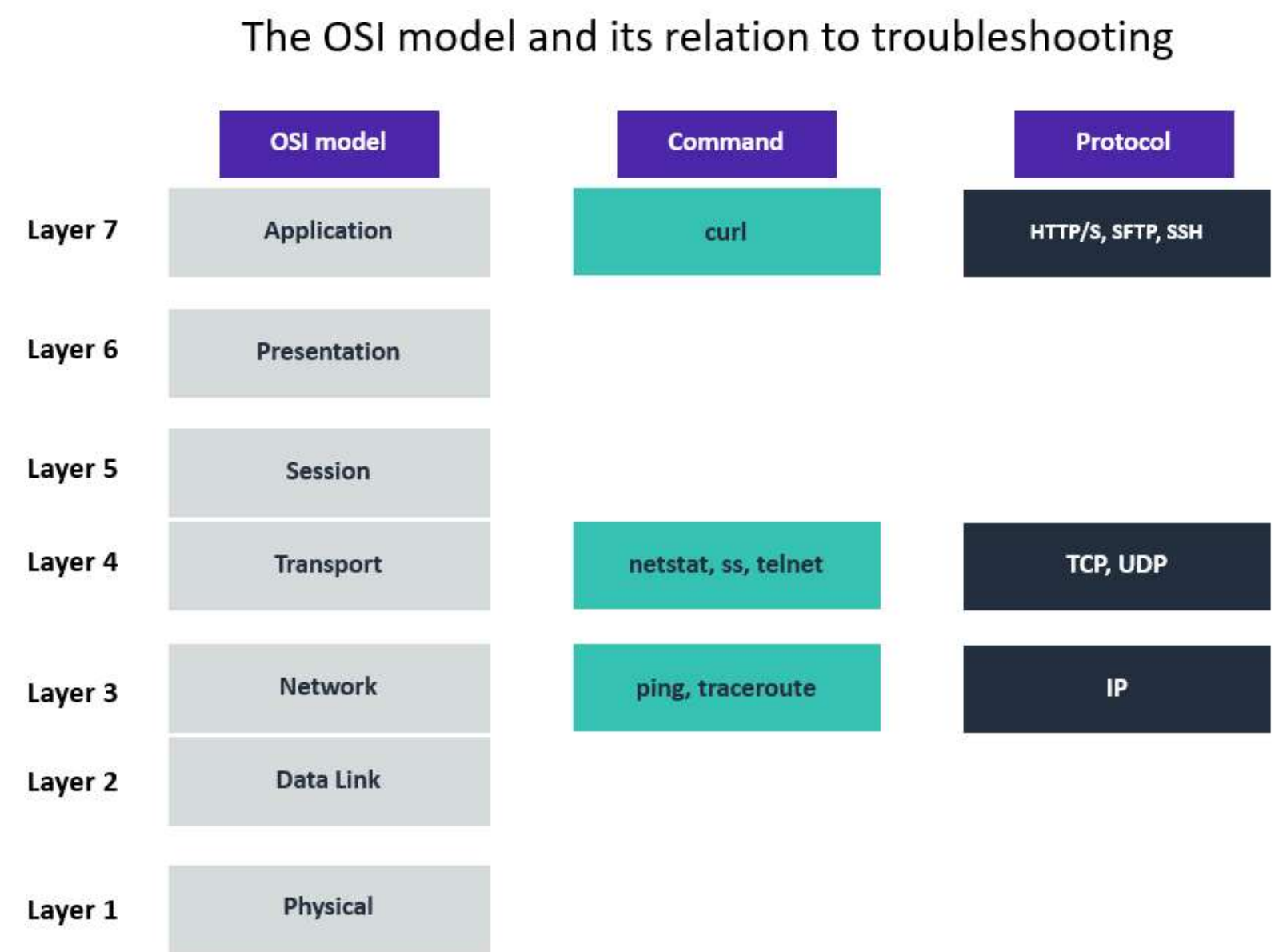
The OSI model and its relation to troubleshooting

OSI model		Command	Protocol
Layer 7	Application	curl	HTTP/S, SFTP, SSH
Layer 6	Presentation		
Layer 5	Session		
Layer 4	Transport	netstat, ss, telnet	TCP, UDP
Layer 3	Network	ping, traceroute	IP
Layer 2	Data Link		
Layer 1	Physical		

Figure: This is an example of how troubleshooting commands have similarities to the OSI model.

Layer 3 (network): The ping and traceroute commands

24. The following is an example of a customer scenario where you can use the **ping** command:

> The customer has launched an EC2 instance. To test connectivity to and from it, run the ping command. You can use this command to test connectivity and ensure that it allows Internet Control Message Protocol (ICMP) requests on the security level, such as security groups and network ACLs.

In the Linux terminal, run the following command, and press Enter:

```
ping 8.8.8.8 -c 5
```

This is the **ping** command. When you run this command, you can input an IP or URL followed by options. In this example, the **-c** stands for count, and **5** stands for how many requests you are requesting.

After you run this command, you should see a result similar to the following:

```
[ec2-user@ip-10-0-10-189 ~]$ ping 8.8.8.8 -c 5
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=93 time=7.81 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=93 time=7.92 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=93 time=7.83 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=93 time=7.82 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=93 time=7.79 ms

--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 7.792/7.837/7.920/0.120 ms
[ec2-user@ip-10-0-10-189 ~]$ []
```

Figure: The ping command shows IP connectivity to a web server.

You can use the ping command for a few reasons, but the most common reason is to test connectivity to something such as a server. The ping command sends ICMP echo requests from your machine to the server that you are trying to reach (for example, amazon.com). The server sends an echo reply with a round-trip time. You use the ping command mostly to troubleshoot connectivity issues and reachability to a specific target. You can also use it to bring a specific network up if traffic needs to continuously flow through a network. You can also send a continuous ping.

25. The following is an example of a customer scenario where you can use the **traceroute** command:

> The customer is having latency issues. They say that their connection is taking a long time, and they are having packet loss. They aren't sure if it is related to AWS or their internet service provider (ISP). To investigate, you can run the traceroute command from their AWS resource to the server that they are trying to reach. If the loss happens toward the server, the issue is most likely the ISP. If the loss is toward AWS, you might need to investigate other factors that might limiting networking connectivity.

In the Linux terminal, run the following command, and press Enter:

```
traceroute 8.8.8.8
```

This is the **traceroute** command. You can input an IP or URL followed by options.

After you run this command, you should see a result similar to the following:

```
[ec2-user@ip-10-0-10-189 ~]$ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  ec2-34-221-151-147.us-west-2.compute.amazonaws.com (34.221.151.147)  3.130 ms ec2-44-233-117-87.us-west-2.compute.amazonaws.com (44.
233.117.87)  6.433 ms ec2-44-233-117-57.us-west-2.compute.amazonaws.com (44.233.117.57)  1.616 ms
 2  100.65.40.176 (100.65.40.176)  3.301 ms 100.65.48.32 (100.65.48.32)  9.307 ms 100.66.16.122 (100.66.16.122)  15.731 ms
 3  100.66.11.160 (100.66.11.160)  18.391 ms 100.66.18.48 (100.66.18.48)  6.892 ms 100.66.36.184 (100.66.36.184)  1.866 ms
 4  100.66.23.112 (100.66.23.112)  5.852 ms 100.66.23.130 (100.66.23.130)  17.152 ms 241.0.2.139 (241.0.2.139)  0.268 ms
 5  108.166.240.17 (108.166.240.17)  0.275 ms 241.0.2.143 (241.0.2.143)  0.267 ms 108.166.240.22 (108.166.240.22)  0.260 ms
 6  242.0.61.1 (242.0.61.1)  0.544 ms 108.166.232.56 (108.166.232.56)  0.277 ms  0.310 ms
 7  242.0.31.1 (242.0.31.1)  0.423 ms 242.0.30.129 (242.0.30.129)  0.245 ms 100.95.1.147 (100.95.1.147)  1.041 ms
 8  100.95.1.153 (100.95.1.153)  1.931 ms 100.95.1.143 (100.95.1.143)  0.964 ms  0.946 ms
 9  100.100.18.44 (100.100.18.44)  0.934 ms 100.100.18.104 (100.100.18.104)  1.006 ms 100.92.37.152 (100.92.37.152)  7.230 ms
10  100.92.37.126 (100.92.37.126)  9.632 ms 100.92.31.148 (100.92.31.148)  7.042 ms 100.92.37.20 (100.92.37.20)  6.833 ms
11  100.92.81.69 (100.92.81.69)  8.084 ms 100.92.82.54 (100.92.82.54)  8.150 ms 100.92.26.85 (100.92.26.85)  10.360 ms
12  100.92.81.71 (100.92.81.71)  10.382 ms 100.92.32.143 (100.92.32.143)  7.843 ms 100.92.82.89 (100.92.82.89)  10.110 ms
13  100.92.29.112 (100.92.29.112)  7.867 ms 100.92.34.110 (100.92.34.110)  8.396 ms 100.92.35.138 (100.92.35.138)  9.412 ms
14  52.93.131.227 (52.93.131.227)  7.333 ms 100.92.34.31 (100.92.34.31)  8.456 ms 100.92.34.49 (100.92.34.49)  7.544 ms
15  52.93.131.228 (52.93.131.228)  11.561 ms 150.222.251.63 (150.222.251.63)  8.510 ms 150.222.244.26 (150.222.244.26)  7.486 ms
16  100.92.127.13 (100.92.127.13)  7.777 ms 100.92.127.98 (100.92.127.98)  16.859 ms 100.92.191.126 (100.92.191.126)  14.912 ms
17  100.92.125.14 (100.92.125.14)  8.773 ms 100.92.191.63 (100.92.191.63)  7.798 ms 100.92.127.63 (100.92.127.63)  8.490 ms
18  100.92.125.49 (100.92.125.49)  8.059 ms 100.92.125.12 (100.92.125.12)  7.193 ms 100.92.125.86 (100.92.125.86)  7.762 ms
19  100.92.125.97 (100.92.125.97)  7.838 ms 150.222.251.141 (150.222.251.141)  10.460 ms 100.92.179.103 (100.92.179.103)  8.705 ms
20  100.92.133.11 (100.92.133.11)  7.453 ms 100.92.133.31 (100.92.133.31)  6.665 ms 100.92.125.27 (100.92.125.27)  8.055 ms
21  52.95.54.205 (52.95.54.205)  9.136 ms 100.92.128.128 (100.92.128.128)  8.600 ms 100.92.128.4 (100.92.128.4)  7.790 ms
22  100.92.128.135 (100.92.128.135)  9.570 ms 99.82.183.167 (99.82.183.167)  6.389 ms 100.92.128.13 (100.92.128.13)  10.012 ms
23  100.92.125.14 (100.92.125.14)  9.496 ms 100.92.125.38 (100.92.125.38)  7.607 ms *
24  108.170.245.97 (108.170.245.97)  10.188 ms * *
25  * 100.92.133.17 (100.92.133.17)  7.807 ms *
26  dns.google (8.8.8.8)  6.439 ms 52.95.53.142 (52.95.53.142)  8.297 ms dns.google (8.8.8.8)  6.902 ms
```

Figure: The traceroute command shows the path taken to the web server and the latency taken to it.

Packet loss, seen as percentages, can occur at each hop, and this loss usually occurs because of an issue with the user's local area network (LAN) or ISP

You can pinpoint an issue or error when the hostnames and IP addresses on either side of a jump have failed. Three asterisks (***) indicate a failed hop.

The traceroute command reports on the path and latency that the packet takes to get from your machine to the destination (8.8.8.8). Each server is called a hop. There can be packet loss, seen as percentages, at each loss, which is usually due to the user's local area network (LAN) or ISP; however, if the packet loss occurs toward the end of the route, then the issue is more than likely the server connection. You can pinpoint an issue or error when hostnames and IP addresses on either side of a failed jump, which looks like three asterisks (***).

Layer 4 (transport): The netstat and telnet commands

26. The following is an example of a customer scenario where you can use the **netstat** command:

> Your company is running a routine security scan and found that one of the ports on a certain subnet is compromised. To confirm, you run the netstat command on a local host on that subnet to confirm if the port is listening when it shouldn't be.

In the Linux terminal, run the following command, and press Enter:

```
netstat -tp
```

This is the **netstat** command. You can use the following options:

- netstat -tp: Confirms established connections

- netstat -tlp: Outputs listening services

  - netstat -ntlp: Outputs listening services but does not resolve port numbers

After you run this command, you should see a result similar to the following:

```
[ec2-user@ip-10-0-10-189 ~]$ netstat -tp
(No info could be read for "-p": geteuid()=1000 but you should be root.)
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address          Foreign Address         State       PID/Program name
tcp        0     64 ip-10-0-10-189.us-w:ssh 205.251.233.178:5477    ESTABLISHED -
```

Figure: The result of netstat -tp confirms established connections.

The netstat command shows the current established TCP connections from which the host is listening. When troubleshooting networking issues starting with the host machine and working outward, you can run this command to understand which ports are listening and which are not. Because this command gives you a snapshot of your layer 4 connectivity, using this command will help you save time when trying to narrow down a large networking issue.

27. The following is an example of a customer scenario where you can use the **telnet** command:

   > The customer has a secure web server and has custom security group rules and network ACL rules configured. However, they are concerned that port 80 is open even though it shows their security settings indicate that their security group is blocking this port, you can run telnet 192.168.10.5 80 to ensure that the connection is refused.

   In the Linux terminal, run the following command, and press Enter to install telnet:

   ```
   sudo yum install telnet -y
   ```

   In the Linux terminal, run the following command, and press Enter:

   ```
   telnet www.google.com 80
   ```

   This is the **telnet** command. You can input an IP or URL followed by the port number to connect to that port.

   After you run this command, you should see a result similar to the following:

   ```
   Installed:
     telnet.x86_64 1:0.17-65.amzn2

   Complete!
   [ec2-user@ip-10-0-10-189 ~]$ telnet www.google.com 80
   Trying 142.251.33.100...
   Connected to www.google.com.
   Escape character is '^]'.
   ```

   Figure: The telnet command confirms the TCP connection to a web server. It makes the HTTP request using telnet.

   The telnet command confirms the TCP connection to a web server making an HTTP request if using port 80 to telnet. You can also use this command at layer 7. If you can successfully connect to the web server, then there is nothing blocking you or the server from connecting. If the connection fails with a message like "connection refused," then something is likely blocking the connection, such as a firewall or security group. If the connection fails with a message like "connection timed out," then then issue may be no network route or connectivity.

Layer 7 (application): The curl command

28. The following is an example of a customer scenario where you can use the **curl** command:

   > The customer has an Apache server running, and they want to test if they are getting a successful request (200 OK), which indicates that their website is running successfully. You run a curl request to see if the customer's Apache server returns a 200 OK.

   In the Linux terminal, run the following command, and press Enter:

   ```
   curl -vLo /dev/null https://aws.com
   ```

This is the **curl** command. You can use the following command options:

-I: This option provides header information and specifies that the request method is Head.
-i: This option specifies that the request method is GET.
-k: This option tells the command to ignore SSL errors.
-v: This option is verbose. It shows what the computer is doing or what the software is loading during startup.
-o /dev/null: This option will send HTML and CSS in response to null.

After you run this command, you should see a result similar to the following:

Figure: The results of the curl command: the output tests the connection to a web service, such as AWS, and submits the HTTP request.

You can use the curl command to transfer data between you and the server. The curl command can use many different protocols, but the most common are HTTP and HTTPS. You can use the curl command to troubleshoot communication from your local device to a server.

Lab Complete 🎓

🏁 Congratulations! You have completed the lab.

29. Select ⌷ End Lab ⌷ at the top of this page and then select ⟨ Yes ⟩ to confirm that you want to end the lab.
    A panel will appear, indicating that "DELETE has been initiated... You may close this message box now."
30. Select the **X** in the top right corner to close the panel.

Recap

▶ **In this lab**

Additional resources

- ping
- telnet
- traceroute

For more information about AWS Training and Certification, see AWS Training and Certification.

Your feedback is welcome and appreciated.

If you would like to share any suggestions or corrections, please provide the details in our AWS Training and Certification Contact Form.