



Conversation with:  
**Colt Anderson**  
Network Administrator

# Network Operations Challenge Chat Transcript

**Colt sent:**

Hey @playerone, it's a pleasure to meet you!

Welcome to the TryCyber Foundation! My name is Colt Anderson, and I work as one of the Network Administrators here.

**Colt sent:**

I'll be your mentor for today, and I'm going to be showing you how to perform some core **Network Operations** tasks. I usually perform my tasks on switches, routers, and servers like the **Ubuntu 22.04 Linux** server you are looking at now.

**Colt sent:**

All the tasks you'll be performing today will require you to use the **Terminal**. If you're unfamiliar with using a terminal, that's fine! I can bring you up to speed with a brief tutorial on it.

**Colt sent:**

Shall we get started?

**Participant sent:**  
Terminal tutorial, please!

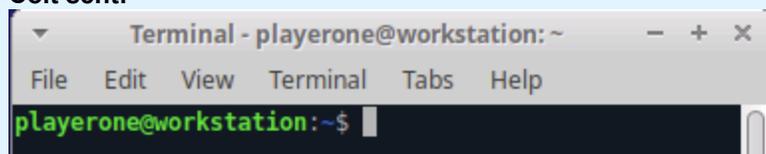
**Colt sent:**

The **Terminal** is primarily used to access an application called a shell. We interact with shells via a **command-line interface (CLI)** where we input and execute text-based commands on a system.

**Colt sent:**

Let's open the Terminal on this system now and run a few basic commands so you get the hang of it. You can open the **Terminal** by double-clicking the icon on the desktop named **Terminal Emulator**. It should look like this once you have it open...

**Colt sent:**



**Participant sent:**  
I've opened the Terminal.

**Colt sent:**

Great. Basically, you type commands into the shell within the Terminal and then hit **Enter** or **Return** on the keyboard to run the command.

**Colt sent:**

To properly utilize any CLI command, you must understand its format, options, and other arguments. The **format** defines the structure and order of options and other arguments, the **options** define or modify behavior, and the **arguments** are command defined input types (e.g., files, directories, users, software packages, etc.).

**Colt sent:**

Additionally, all CLI commands on this system are case-sensitive, so pay close attention to the capitalization of letters.

**Colt sent:**

The basic command format is...

```
command [OPTIONS] [ARGUMENTS]
```

However, it varies a lot from command to command, and options and other arguments are not always required.

**Colt sent:**

An example of a command that does not require any options or other arguments is `whoami`. If you type `whoami` into the shell and then hit `Enter` or `Return` on the keyboard, the `whoami` command will output the name of the user running the command into the Terminal. It should look like this if you run that command...

**Colt sent:**

```
playerone@workstation:~$ whoami  
playerone
```

**Participant sent:**

What's a more involved example?

**Colt sent:**

I'll give you a more complex example using the `ls` command.

**Colt sent:**

In this example, we will use the `ls` command to list detailed information about the entire contents of the `Templates` directory (i.e., folder) in your (playerone's) `home` directory. Our example `ls` command will use the following format...

```
ls [OPTIONS] [DIRECTORY]
```

Note that in this case, the command's argument requires the input to be a directory.

**Colt sent:**

The actual command we want to run is...

```
ls -a -l /home/playerone/Templates
```

**Colt sent:**

The `-a` and the `-l` are both options that modify the `ls` command's behavior, and `/home/playerone/Templates` is our argument which is a directory provided in the form of a path.

**Colt sent:**

For this command's argument, we must provide the **path** to the directory from the root of the file system; otherwise, the system won't know which directory named `Templates` we are referring to.

**Colt sent:**

And finally, when you run that command, the output should look like this...

**Colt sent:**

```
playerone@workstation:~$ ls -a -l /home/playerone/Templates/
total 24
drwxr-xr-x  2 playerone playerone 4096 Jul 12 21:36 .
drwxr-x--- 14 playerone playerone 4096 Jul 12 21:36 ..
-rw-r--r--  1 playerone playerone 6859 Apr  9 2022 'OpenDocument Spreadsheet.ods'
-rw-r--r--  1 playerone playerone 7388 Apr  9 2022 'OpenDocument Text.odt'
-rw-r--r--  1 playerone playerone    0 Apr  9 2022 'Plain Text.txt'
```

**Participant sent:**  
Got it. Any extra notes?

**Colt sent:**

The Terminal and shells are incredibly powerful and versatile tools, and not all commands and programs follow the general structure I've provided you with here. Unfortunately, we only have time to cover the basics, but I do have a few last things I'd like to mention.

**Colt sent:**

Some commands will not print visual output to the Terminal in normal operation, such as the `cp` command, which is used to copy files and directories.

**Colt sent:**

Many commands can use multiple arguments, handle multiple input types for arguments (e.g., file and/or directory paths), or have options that will have their own arguments.

**Colt sent:**

Some options can, or must be, written in a long-form format (e.g., `ls --all [DIRECTORY]` is the same as `ls -a [DIRECTORY]`).

**Colt sent:**

Options without arguments can often be provided together and in any order (e.g., `ls -la [DIRECTORY]` is the same as `ls -a -l [DIRECTORY]`).

**Colt sent:**

And last, but certainly not least, you can almost always reference a command's format, options, and other arguments using the command `man [COMMAND]` (e.g., `man cp`) to view the provided command's manual page in the Terminal.

**Colt sent:**

Hopefully that was not too much! I know it seems like a lot, but it gets easier the more you use it. For today's tasks, I'll be sure to provide you with more details for any commands and programs you'll need.

**Participant sent:**

Sounds good! I'm ready to get started!

**Colt sent:**

Great!

**Colt sent:**

Today, we will be assisting one of our development teams in integrating a server onto the network so they can do some specialized A/B testing. A/B testing compares two (although in this case it will be three) versions of software to see which one performs better.

**Colt sent:**

The infrastructure team has provisioned the server, which is the Ubuntu 22.04 Linux server you are looking at now, and now the network operations team needs to integrate the server into the network per the development team's request.

**Colt sent:**

You may have guessed this, but you and I are the network operations team today. 🧐

**Colt sent:**

Making sure things get integrated into the network properly and maintaining the network are core Network Operations duties!

**Colt sent:**

Today's tasks, as per the development team's request, will be **adding two temporary IP addresses to this server's main network interface**, sometimes called a NIC, for their A/B testing. IP addresses are assigned to every device on an IP-based network and allow those devices to identify, locate, and communicate with each other.

**Participant sent:**

Good to know! Where do we start?

**Colt sent:**

To start off, let's **review the existing IP addresses configured** on the server's primary NIC, `ens3`. To see what IP addresses are currently configured on `ens3`, run the following command in the Terminal...

```
ip -4 -br address show
```

**Colt sent:**

That `ip` command uses the `-4` option to limit the output to only include IPv4 IP addresses and the `-br` option to further limit the output to just basic details. Then the `address` object followed by the `show` option indicates we want to see the addressing information for the server's network interfaces.

**Colt sent:**

When you run that command, you should see the following output...

**Colt sent:**

```
playerone@workstation:~$ ip -4 -br address show
lo                UNKNOWN          127.0.0.1/8
ens3              UP               192.168.100.200/24
playerone@workstation:~$
```

**Participant sent:**

Got it.

**Colt sent:**

Let's take a look.

**Colt sent:**

Ah, as expected, `ens3` only has one IP address assigned to it, which is `192.168.100.200` with a subnet mask of `24`. This IP address is permanently set by the server's netplan configuration.

**Colt sent:**

We need to add two additional, but temporary, IP addresses to the `ens3` NIC, which will result in `ens3` having three IP addresses in total.

**Colt sent:**

While all three IP addresses will show up the same in the output of the `ip` command we ran just a minute ago, the temporary ones will disappear if the server is rebooted. Temporary IP addresses are generally used for short-term stuff, like testing or network migrations.

**Colt sent:**

Alright, let's **add the first of those two temporary IP addresses!** The first IP address we need to add to this server is `192.168.100.210` with a subnet mask of `24`. To add the first IP addresses, run the following command in the Terminal...

```
sudo ip addr add 192.168.100.210/24 dev ens3
```

The above command utilizes the `sudo` command, which will sometimes prompt you for your password. Just enter your password into the prompt and hit `Enter` or `Return` on the keyboard when you're done. (Note: Your password can be found on the Info Tab)

**Colt sent:**

The `ip` command shows and manipulates various network settings, including NIC modifications. You may have noticed the `ip` command structure is a little different than other commands, which is because it is formatted more like a networking device's (e.g., a router) CLI.

**Colt sent:**

The `ip` command I just provided you uses the `addr` object followed by the `add` action to specify that we want to add an IP address, which is then followed by `192.168.100.210/24` which is the IP address we want to add formatted in [IP]/[Subnet] format. The IP address is then followed by `dev`, short for device, and `ens3`, the name of the NIC we want to add the IP address to.

**Colt sent:**

If you input and run a valid `ip addr add ...` command, it will not return any output to the Terminal. Let me know once you've added the first temporary IP address.

**Participant sent:**  
I've run that command!

**Colt sent:**

Awesome!

**Colt sent:**

Now, let's re-run that first `ip` command I gave you to **confirm that we've added the correct IP address**. That command was... `ip -4 -br address show`

**Colt sent:**

You should now be looking at the following output, which would confirm you have successfully added the first IP address...

**Colt sent:**

```
playerone@workstation:~$ ip -4 -br address show
lo                UNKNOWN          127.0.0.1/8
ens3              UP               192.168.100.200/24 192.168.100.210/24
playerone@workstation:~$
```

**Participant sent:**  
That is the output I got.

**Colt sent:**

Fantastic!

**Colt sent:**

Anddd... I just got a high-priority task assigned to me. 😬

**Colt sent:**

So... it looks like I'll need you to handle **adding the second and last temporary IP address** on your own.

**Colt sent:**

But don't worry! **The process to add the second temporary IP address is identical to the process you used to add first IP address**, except the **second IP address** we need to add is `192.168.100.220` with a subnet mask of `24`.

**Colt sent:**

You can verify that you've successfully added the second temporary IP address by running the following command again...

```
ip -4 -br address show
```

**Colt sent:**

Which should have output that looks like this once you've done everything correctly...

**Colt sent:**

```
playerone@workstation:~$ ip -4 -br address show
lo                UNKNOWN          127.0.0.1/8
ens3              UP               192.168.100.200/24 192.168.100.210/24 192.168.100.220/24
playerone@workstation:~$
```

**Colt sent:**

Alright, I've got to run now! Thank you for all your help today! 🙌

TRY  
CYBER

© 2023 Try Cyber - Sponsored by CISA