Hands-on Lab: Installing, Updating, and Working with Text Editors

Estimated time needed: **45** minutes

Learning Objectives

After completing this hands-on lab, you will be able to:

- Use the `sudo` command to enable access to "super-user" system administration tools
- Use the `apt` system administration command to update and install two popular packages for text editing: **nano** and **Vim**
- Create and edit files using nano
- Create and edit files using Vim

About Skills Network Cloud IDE

Skills Network Cloud IDE (based on Theia and Docker) provides an environment for hands-on labs for course and project-related labs. Theia is an open source Integrated Development Environment (IDE) that can be run on the desktop or on the cloud. To complete this lab, you will be using the Cloud IDE based on Theia.

Important notice about this lab environment

Recall that sessions for this lab environment are not persistent, so any data or files you may have saved in a previous session will be lost. Please plan to complete these labs in a single session.

Exercise 1 - Upgrading and installing packages

In your lab environment, we provide you access to a system admistration utility called "super-user do", or `sudo`.

> Although this sandbox environment will not require a password to run the `sudo` command, typical Linux production-oriented environments will require a password to run `sudo`.

You will need to use the `sudo` command to activate the powerful `apt` command. You'll use `apt` to upgrade nano to its latest version. You will also use `apt` to install Vim.

> `apt` (*Advanced Packaging Tool*) is a powerful command line tool. You use it to perform system administration operations such as installing software packages, upgrading existing packages, and updating your system's *package list index*.

1.1 Updating your Linux sytem's package list index

Before installing or upgrading any packages on your Linux system, it's best practice to first `update` your package list index. Go ahead and enter this command,

1. 1

1. sudo apt update

Copied!

to update (re-synchronize) your package index files from their sources. This will take a bit of time to run. While you're waiting, if you're interested, go ahead and open another terminal and view the locations `apt` uses to access those sources in the file `/etc/apt/sources.list`.

In short, running `apt` with the `update` option ensures all of your package dependencies are up-to-date and correctly specified prior to making any changes to your system's packages.

1.2. Upgrading nano

nano is a simple *command line editor* that enables you to use the terminal window as a text editor.

nano is already installed on your system. Go ahead and upgrade to the latest supported version of nano by entering:

1. 1

1. sudo apt upgrade nano

Copied!

You may be prompted: `Do you want to continue? [Y/n]`

Type `Y` and press `Enter` to continue. Updating nano will take some time to complete.

> The capital `Y` in `Y/n` (yes or no) means "yes" is the default - if you press Enter without typing anything, the terminal assumes you are choosing "yes".

Note: If you encounter a colon (:) similar to the one highlighted in the screenshot while upgrading Nano, please press 'q' to exit and continue with the upgrade process.

You'll get to use nano soon, but first, let's take a look at the other main use case for `apt`, installing packages.

1.3. Installing Vim

Another popular text-editing program is **Vim**. Vim is a highly configurable text editor built for efficiency. It takes some practice to get good at using Vim, but the time investment is very worthwhile.

Because Vim isn't preinstalled on your Linux system, you'll need to install it yourself. If you haven't already done so in this session, ensure you run the command `sudo apt update`. Then to install Vim, enter the following command:

1. 1

1. `sudo apt install vim`

[Copied!]

Similar to when you updated nano, you may again be prompted: `Do you want to continue? [Y/n]`

Type `Y` and press `Enter` to continue. Vim will begin installing on your system.

In a few exercises, you will use Vim to edit a text file.

Exercise 2 - Creating and editing files with nano

In this exercise, you will use the command line editor nano to create and edit a file.

> nano is known for being a simple and easy-to-master text editor. Vim is harder to learn, but it has many expert-level features that nano doesn't offer.

2.1 Navigating to the project directory

We provide you with an empty project directory at `/home/project`. Ensure you're working in this folder by changing directories using the command:

1. 1

1. cd /home/project

Copied!

Try auto-completing the path by typing `cd /home/pr` and pressing the `Tab` key.

If you enter `ls` here, you shouldn't see any files or subdirectories listed.

2.2 Creating and editing a text file with nano

To create a new file, enter

1. 1

1. nano hello_world.txt

Copied!

in the terminal. This will simultaneously create a new file called `hello_world.txt` and enable you to begin editing it using the nano text editor.

> Double-check that your new file was created by opening another terminal window and running the `ls` command on `/home/project`. You should see `hello_world.txt` listed.

In your nano terminal, whatever you type will be added to your *text buffer*, where text is stored until you save it. Type the following text in your nano terminal:

1. 1

1. Hello world!

Copied!

This will create the text `Hello world!` in your text buffer.

To create another line of text, press `Enter`. In your new line, type

1. 1

1. This is the second line of my first-ever text file created with nano.

Copied!

to create a second line of text in your text buffer.

Now:

1. Press `CTRL+X` to exit nano.

2. You will be prompted as follows:

1. 1
2. 2
3. 3

1. Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
2. Y Yes
3. N No   ^C Cancel

Copied!

Press Y to save your new lines of text to your file.

    3. Press Enter to confirm the file name.

At this point, nano should have exited and returned you to the command prompt.

2.3 Verifying your new text file

By entering a familiar command, such as

1. 1

1. cat hello_world.txt

Copied!

you should be able to inspect and verify that your new file contains the two lines you wrote to it with nano. Cool!

Exercise 3 - Creating and editing files with Vim

3.1 Quick intro to Vim

Recall that Vim has two basic modes: Insert mode, where you enter text, and Command mode, where you do everything else. You can start Vim simply by entering

1. 1

1. vim

Copied!

at the command prompt, which displays something like the following in your terminal window:

Notice that you can get help on Vim by entering `:help`, and you can quit Vim by entering `:q`.

Go ahead and enter `:help`. This brings up an informative help file you can scroll through.

When you are done reading, simply enter `:q` to quit Vim and return to the command prompt.

Vim is very powerful and takes some time to learn. We're just covering the very basics here and leaving it to you to explore further. Check out the official Vim site at https://www.vim.org.

3.2 Creating and editing a text file with Vim

Begin by navigating back to your `/home/project` directory if you aren't already there.

Type the command,

1. 1

1. `vim hello_world_2.txt`

Copied!

to create a new file called `hello_world_2.txt`and edit it using Vim.

Once your Vim session has started, go ahead and press `i` to enter `Insert` mode. This is the mode where you can enter and delete text in the text buffer.

Go ahead and type some text in the buffer, for example:

1. 1

1. `Hello World!`

Copied!

Just like in nano, press `Enter` to start a new line, and then type

1. 1

1. `This is the second line.`

Copied!

to create a second line of text in the buffer.

When you're done typing text in the buffer, press the Escape key, `Esc`, to exit the `Insert` mode. This brings you to `Command` mode.

> In Vim, it's easy to accidentally end up in a mode you didn't intend to be in. No worries - you can use the Esc key to return to Command mode.

Now that you are back in `Command` mode, you can save your work by entering the command `:w`. This writes the contents of the text buffer to your text file.

Finally, you can exit your Vim session by entering `:q`.

Practice Exercises

1. Using nano, edit your new `hello_world.txt` file to add a new line containing the following text:

1. 1

1. `This is line three of my new file.`

Copied!

Then, save your changes and exit nano.

> **Tip**: Use the `Up Arrow` key to go through your command history until you see the right command.

▼ Click here for Solution

> Begin editing the existing file `hello_world.txt` with nano:

1. 1

1. `nano hello_world.txt`

Copied!

> Type the following content in the editor:

1. 1

1. `This is line three of my new file.`

Copied!

> Then save and exit using:

1. Press `CTRL` + `x` to exit
2. You will be prompted to (save y/n?). Press `y` to save.
3. Press `Enter` to confirm the file name.

2. Using Vim, create a file called `done.txt` that prints `"I am done with the lab!"` when you execute the file with Bash.

In this exercise you will use Vim to create a file that contains the `echo` command, which you may not have seen yet. You will also *run* the file using Bash, a shell scripting language that you will learn more about later in this course. Basically, the file you create will contain a basic command that Bash can interpret and do something with.

▶ Click here for Hint
▼ Click here for Solution 1

> Enter:

1. 1

1. `vim done.txt`

Copied!

▼ Click here for Solution 2

> Press `i` to enter `Insert` mode

> Add the following content to the file:

1. 1

1. `echo "I am done with the lab!"`

Copied!

> Then save your work and exit by:

1. Using `Esc` to exit out of `Insert` mode and return to `Command` mode
2. Typing `:wq` and presssing `Enter`

▼ Click here for Solution 3

Once you've saved and exited and are back at the command prompt, you can run the file with Bash by entering:

1. 1

1. bash done.txt

[Copied!]

This command invokes the Bash shell to interpret the text contained in `done.txt` as a command. In particular, Bash runs the `echo` command along with the quoted text as the input to `echo`, while `echo` simply prints the input text to the terminal window.

When finished, you should see the following text echoed to your terminal:

1. 1

1. I am done with the lab!

[Copied!]

Summary

Congratulations! You've just gained a lot more hands-on experience with the Linux terminal!

By now you are beginning to understand a lot more about how your Linux system functions. In this lab, you learned how to:

- Perform some fundamental **sys admin** operations, such as updating your package list, upgrading existing packages, and installing new packages
- Create and edit a few text files using some serious command-line text editors, nano and Vim
- Understand how the Bash scripting language can be used to interpret commands you include within a text file

Keep up the great work! We still have many more exciting things for you to explore, and we hope your enthusiasm for Linux is growing!

As usual, if you ever feel like you might have missed anything important or just want some extra practice, you can always return to this lab again. Explore and experiment to your heart's content.

Authors

Jeff Grossman
Sam Prokopchuk

Other Contributors

Rav Ahuja

© IBM Corporation. All rights reserved.