Hands-on Lab: Introducing Linux Terminal

Estimated time needed: **40** minutes

Objectives

In this lab, you will:

- Interact with the Linux Terminal
- Navigate directories on a Linux filesystem and explore their contents
- Install and update packages
- Create and edit files using `nano`
- Run shell commands and applications from the terminal

About Skills Network Cloud IDE

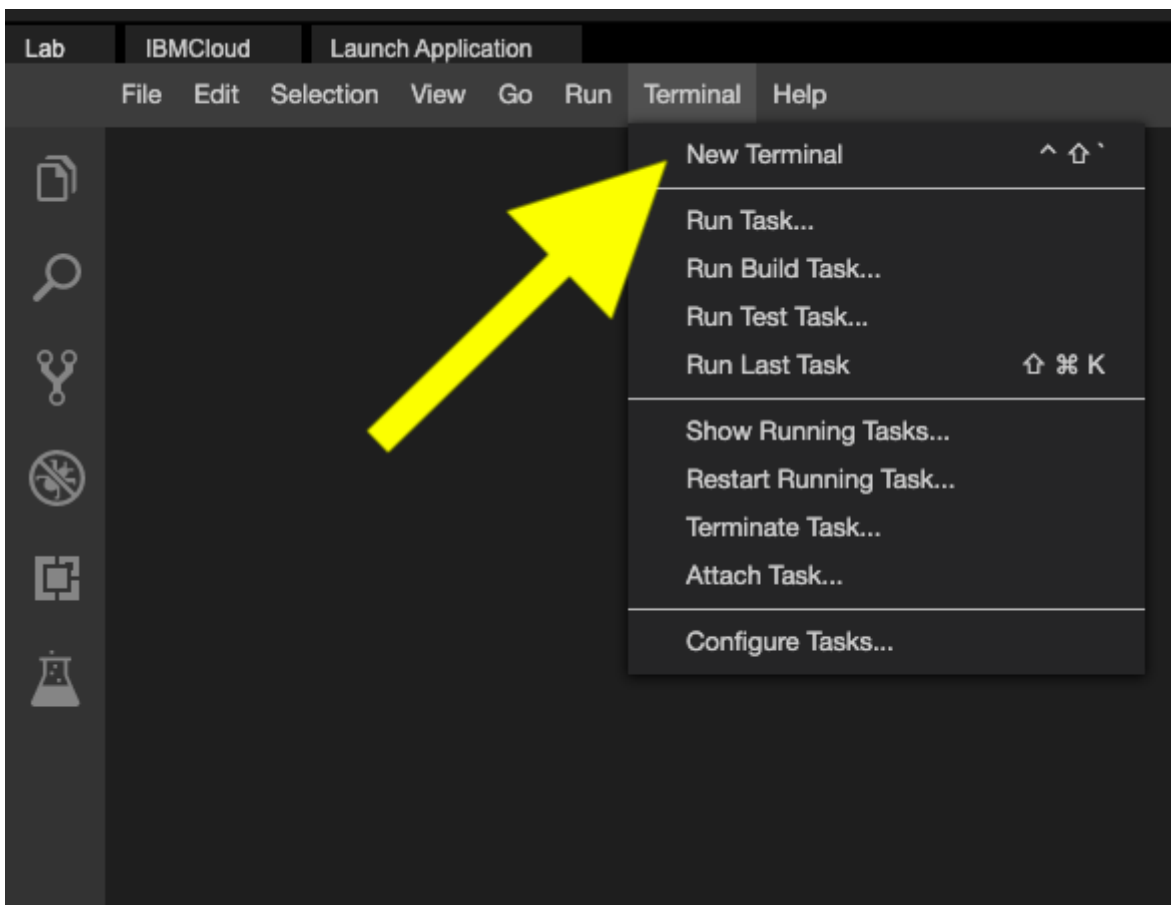
Skills Network Cloud IDE (based on Theia and Docker) provides an environment for hands-on labs for course and project-related labs. Theia is an open source Integrated Development Environment (IDE) that can be run on the desktop or on the cloud. To complete this lab, you will be using the Cloud IDE based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

Important Notice About This Lab Environment

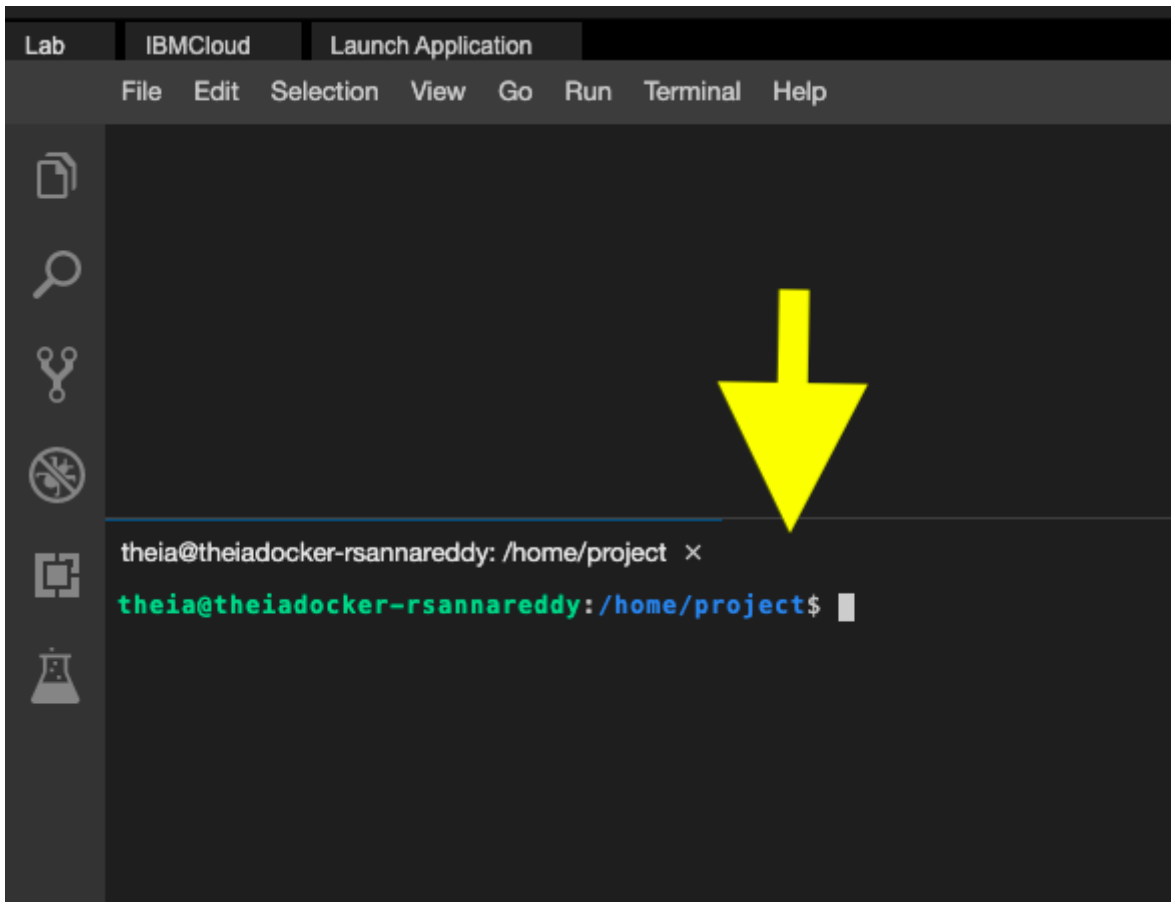
Please be aware that sessions for this lab environment are not persisted. Thus, every time you connect to this lab, a new environment is created for you and any data or files you may have saved in a previous session will be lost. To avoid losing your data, plan to complete these labs in a single session.

Setup

Open a new terminal by clicking the menu bar and selecting **Terminal**->**New Terminal**.



This will open a new terminal at the bottom of the screen.



You can run the commands provided in the following excercises in your newly opened terminal. You can copy the code to your clipboard by clicking the copy button on the bottom right of each codeblock, and then pasting it on the command line.

Exercise 1 - Navigating Directories

`cd`

In this exercise, you will explore directories on the cloud IDE Linux system using the `cd` command.

Recall the special paths:

Symbol	Stands for
`~`	Home directory
`/`	Root directory
`.`	Current directory
`..`	Parent directory

1.1. Changing working directory to home directory

```
cd ~
```

Copy the command above into the terminal and press Enter to run the command.

This will change your current working directory to the home directory `~`.

Note: (In our lab environment, your user's home directory is `~` = `/home/theia`).

1.2. Changing working directory to parent

```
cd ..
```

This will change your current working directory to the parent of the current working directory.

If your working directory was `/home/theia`, then it will become `/home`.

1.3. Changing working directory to root directory

```
cd /
```

This will change your current working directory to the root directory `/`.

1.4 Changing working directory to child

```
cd bin
```

This will change your current working directory to the `/bin` directory.

The `bin` directory is called a *child* of the root `/` directory because it's inside of it.

1.5. Changing working directory back to home directory

```
cd ../home/theia
```

This will change your current working directory back to your home directory.

Of course, a simpler way to do this would be:

```
cd ~
```

1.6. Changing working directory back to project directory

```
cd ../project
```

This will change your current working directory back to your project directory.

The project directory is a special empty directory we provide for your work.

Exercise 2 - Browsing Directories

`ls`

In this exercise, you will explore browsing the content of directories using the `ls` command.

`ls` is a special command that the shell knows by default. You will learn about many more of these commands in the future.

2.1. Viewing files in the current working directory

```
ls
```

Typing `ls` by itself will show all files inside the current working directory.

Because you're in the `/home/project` directory (which is empty) ls will return nothing.

2.2. Viewing files in any directory

If you know the path to a directory, you can view its contents by typing:

```
ls [PATH TO DIRECTORY]
```

For example:

```
ls /
```

This will show the contents of the root directory.

Recall some of the directories you've learned in prior video(s):

Directory	Contains
`/bin`	System libraries
`/sbin`	Binaries that require root privileges
`/usr`	User programs and data
`/home`	Home directory
`/media`	Removable media device directories

```
ls /bin
```

This will show the contents of the `/bin` directory.

You might notice one of these files is called `"ls"`. That's because the `ls` command runs using the file `/bin/ls`.

Exercise 3 - Updating and Installing Packages

In your lab environment, we provide access to the `sudo` command. Be careful not to break your system!

3.1 Getting latest packages information

```
sudo apt update
```

This will fetch the lastest package information from trusted sources.

`apt update` doesn't actually update your packages; instead, it finds if any packages *can* be upgraded.

3.2. Updating nano

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

To get the latest supported version of `nano`, type:

```
sudo apt upgrade nano
```

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

Type "`n`" and press Enter to continue. Updating `nano` will take time and will not affect this lab.

> Note: The capital `Y` in `Y/n` means it's the default - if you press enter without typing anything it uses the default `y`.

3.3. Installing vim

Another popular text-editing program is `vim`.

Because `vim` doesn't come with your system, you will need to install it:

```
sudo apt install vim
```

As with upgrading, you may be prompted: "`Do you want to continue? [Y/n]`"

In this case, type "`y`" and press Enter to continue. You will be using `vim` in a later exercise.

Exercise 4 - Creating and Editing Files

For the purpose of this lab, you will be use `nano` to create and edit files.

This is because `nano` is known as simple to use and easy to master.

On the other hand, vim can be harder to learn - though it has many more features.

4.1 Navigating to the project directory

We provide a clean project directory at `/home/project`. Ensure you're working in this folder using:

```
cd /home/project
```

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

If you type `ls` here, you should see no files.

4.2 Creating and editing a file

```
nano myprogram.py
```

This will create a `.py` (Python) file called `myprogram.py` and enable you to begin editing it using the `nano` text editor.

Type the following to the file:

```
print('Learning Linux is fun!')
```

Now:

1. Press "CTRL-X" to exit

2. You will be prompted with:

```
Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
 Y Yes
 N No           ^C Cancel
```

Press "y" to save.

3. Press "ENTER" to confirm the file name.

You should now be back at the terminal's command prompt.

4.3 Running the Python file you made

```
ls
```

You should now see that the file `myprogram.py` was created in your current working directory.

You can now run your Python file using:

```
python3 myprogram.py
```

Try auto-completing the command by typing `python3 my` and pressing TAB.

You should see the output:

```
Learning Linux is fun!
```

Otherwise, you may have had a typo in your program.

Practice Exercises

1. Problem:

> *Display the content of the `/usr` directory.*

▼ Click here for Solution

```
ls /usr
```

2. Problem:

> *Navigate to the `/media` directory.*

▼ Click here for Solution

```
cd /media
```

3. Problem:

> *Navigate to the `/home/project` directory and display its contents.*

▼ Click here for Solution

```
cd /home/project
ls
```

4. Problem:

> *Using `nano`, edit `myprogram.py` to add a new line containing "`print('My name is ...')`" (replace ... with your name)*

Hint: To start, you can press the Up arrow until you get `nano myprogram.py` from your command history.

▼ Click here for Solution

> Begin editing the existing file `myprogram.py` with nano:

```
nano myprogram.py
```

> Add the following contents to the file:

```
print('My name is [ENTER YOUR NAME HERE]')
```

> 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.

> Once back to the command prompt, you can run the file using:

```
python3 myprogram.py
```

5. Problem:

> *Using `vi`, create a file called "`done.py`" that prints "I am done with the lab!"*

▼ Click here for Hint

> Use `vi` and python's `print` function.

▼ Click here for Solution

> Create and begin editing file `done.py` with `vi`:

```
vi done.py
```

- Press "i" to enter "Insert mode"

> Add the following contents to the file:

```
print('I am done with the lab!')
```

> Then save and exit using:

1. ESC to exit out of "Insert mode"
2. type ":wq" and press ENTER

> Once back to the command prompt, you can run the file using:

```
python3 done.py
```

Authors

Sam Prokopchuk

Other Contributors

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2022-04-05	1.0	Sam Prokopchuk	Add first iteration of lab