***Let’s explore the concepts as part of Section 1.***

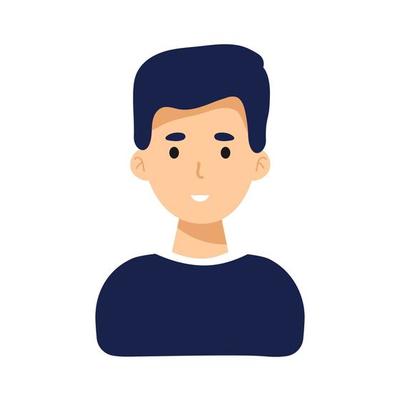
* Fundamentals of Linux [1]
* Linux Components [2]
* Linux Distributions [3]
* Linux Setup [4]
* Package Managers [5]

Let’s start with the absolute basics, that is Operating System.

**Instead of Learning What is Operating System, Let’s start with why Operating systems.**

WHY?

OS

CLI / GUI





This might take back to your school where you learned basic computer education…

You might have learned “The physical component of any computer is what we call as hardware”.

What is physical component? à Combination of CPU, Memory, Storage, Motherboard and network interfaces.

There are two primary actors, who wants to interact with the hardware or who wants to interact with the computer. One is the users, that is you and me. Two, software applications.

Users want to interact with the hardware may be to create files on the computer, folders on the computer, or even to manage the resources of the hardware. May be to check the status of the cpu or memory.

Applications: For example, you are watching this class on Microsoft Teams recording which is nothing but a software application. And for Microsoft Teams to run it needs CPU, it needs Memory and it also needs some disk space. And all of these are available with the hardware.

*So, both users and application want to interact with the hardware. However, both of them are not directly capable of interacting with the hardware.*

If you take applications first, imagine you write a simple hello-world program in Python. You do not write any instructions on how to manage the CPU for this program; you didn’t write any instructions on how to get memory allocated for this program. So, obviously your application cannot deal with the hardware resources because you have not written the code. Similarly, as users, you cannot directly interact with the physical component.

You at least need a graphical user interface or command line interface. And both of them are not provided by the hardware. Because of these reasons, both of these actors (users & Application) cannot interact with the hardware. But a intermediate layer which is called as Operating System can help them.

**What is Operating System?**

Operating System is a special software program that knows how to use the hardware.

So, it can basically perform process management, memory management, device management and even network management.

Because application is also a software program. It can talk to the operating system which is a software layer and this operating system depending on the program it allocates the CPU it allocates the memory and required hardware resources.

This is how your programs actually interact with the hardware or software interact with the hardware through operating system.

Now, you might wonder okay but as users, we cannot still interact with the operating systems. Yes, but what operating systems do specially for the users they (operating systems) provide an additional layer which is CLI or GUI.

If you take windows Operating system, it provides a rich GUI, whereas if you take Linux operating system, it provides rich CLI. Anyways, through this layer (GUI/CLI) users interact with Operating System and in turn talk to the hardware resources.

*Now if you look at the definition of Operating System! Operating System is a n intermediate software layer that acts as a bridge between software and hardware or acts as a bridge between user and hardware. It performs process management, memory management, device management and network management.*

**Note: -**

You can find this definition in the GitHub repository <https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/blob/main/01-getting-started/01-fundamentals.md> as well, where I have explained more about process

management, memory management and other thigs that your Linux Operating System can do.

Now seeing this, Linux is not the only operating system. There are other operating systems like Windows, Mac OS and multiple other. However, these are the three popular Operating Systems in the market.

At this point of time, it’s worth understanding the history of Operating Systems. So, lets take a sneak peek.

In 1960’s, Unix ix the first operating system that was developed.

Before Unix, it was very difficult for the application developers. Although the application developers were very very less, still it was difficult for them to interact with the hardware.

They used to do some advances programming or typical hardware level coding to get their applications interact with the hardware.

And Unix solved this problem. So, it became very popular in no time. And looking at the success of Unix, in 1970’s another operating system was released which is Minix. Minix was kind of developed on top of Unix; Unix was Open Source but Minix was not completely Open Source. It was partially Open Sourced.

In 1980’s there was a revolution in space of Operating Systems when Microsoft released their Operating System which is Windows. And Microsoft’s Operating System unlike the previous operating systems had a rich graphical user interface.

Now, this rich Graphical User Interface (GUI) made communications to the users with the hardware super simple. They (Users) don’t even need to know the commands which was required with Unix as well as Minix.

In 1990’s, because windows was very successful, Minix was also very successful. However, they both were kind of proprietary, they (users/ organizations) need some licenses for the developers to use very specifically. That’s where in 1990’s a person called **Linus Torvalds** worked on the development of Linux kernel and this was the first popular open-source operating system. Although Unix was also open-source operating system, this (Linux) was competitive with windows or in many of the parameters it also surpassed Windows.

So, Linux became the new popular choice for the developers and also for the companies. Even today, that is in 2025.

If you look at Linux versus Windows, 90% of the production workloads today, whether it is software running on your mobiles or the applications that you are using deployed on the servers, most of them, 90% of them use Linux OS.

It’s not because Linux is free. Many people think because Linux is free, companies go for Linux.

This is some part true. However, the major part is that Linux is open-source. Now, because Linux is opensource, it’s not backed by a single company, but it is backed by a community of developers.

So, these developers they are huge in number and due to the active contributions, they ensure Linux is always secure. So, the major reason why companies go for Linux over Windows is that Linux is ***Open-Source***, it is ***free*** and it is ***super secure*** compared to other Operating Systems. And this combination is very rare for companies, something that they get for free and very secure.

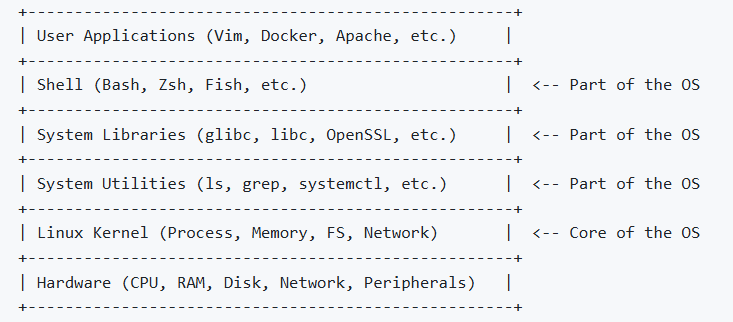
So, companies grab that opportunity. That’s why Linux became a very popular Operating System.

***Recap:***

Let’s do a very quick recap. At this point of time, we learned what is operating system.

* Operating System is an intermediate software layer between your software applications or users and the hardware.
* It is the heart of your computer.
* There are multiple Operating Systems such as Windows, Linux and Mac OS.
* We also looked at the history of Operating System and learned that Linux is the popular choice when compared to Windows.

Now that we have learned the fundamentals of Linux or Operating System. Let’s try to understand what is the structure of Linux [2]



So, the main component of Linux Operating System is called as Kernel.

So, what does Linux ***Kernel*** do?

Everything that we discussed till now, so the interaction with hardware such as process management, memory management, disk management and network management is handled through the Linux kernel.

So, this (Kernel) is basically like the engine of the car which is doing the heavy lifting.

On top of Linux Kernel, you know there are some layers like system libraries or some system packages and on top of that there is a Command Line Interface (CLI), or at times Linux also provide Graphical User Interface (GUI).

***Note: -***

You can also find that here in if you go to linux-structure in GitHub repo <https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/blob/main/01-getting-started/02-linux-structure.md> , you will notice

* The physical component is ***Hardware***.
* On top of that, there is something called as ***Linux Kernel*** which takes care of Process management, Memory Management, File System management and Network management.
* To make the Linux Operating System rich, now it also comes with ***System Utilities*** such as basic commands out of the box and some basic libraries out of the box. So, this ensures that Linux is rich in features and also Linux is secure.
* On top of that, there is binaries such as CLI or GUI. Usually, when it comes to linux, its CLI.

That’s it! So, these four things is basically what we call as Linux Operating System. And the most important component of Linux Operating System is Linux Kernel.

***Thought Provoking:***

**Hey Devender!**

**Is it possible to ship Linux Operating Systems without these things (System Utilities & System Libraries & Shell)?**

You can still do it today, because Kernel is the one which is taking care of the major responsibility, so you can actually remove most of these things (System Utilities & System Libraries & Shell) and even ship a Linux Operating system.

But this (System Utilities & System Libraries & Shell) will help the users, that’s why you have System Utilities, System Libraries & Shell. Shell is mandatory. Without CLI people/users cannot interact with the hardware.

Now we understood

* The Operating System of linux.
* The Components of Linux

Now let’s see what exactly is this Linux Distributions [3]

You know, we keep hearing about Ubuntu, Red Hat, Fedora, Debain, Alpine…

What exactly are all of these things?

You know it’s a very very easy concept.

As I told you, Linux is open-source, so the people ***Linus Torvalds*** and other contributors, they have developed Linux and they made it open source.

So, when it is open source, you are free to download the code and you can add more features to it or remove certain features to it and ship it as a product. That is where companies line [***Canonical***](https://canonical.com/)saw the opportunity.

What they do even today [***Canonical***](https://canonical.com/)as a company, they take the copy of the open-source Linux, they add some more features, they add more wrappers on top of Linux Operating system and provide that as Ubuntu Linux distribution.

There is another company called [***RedHat***](https://www.redhat.com), also does the same thing. So, they take the copy of the Linux Operating System code, they add more wrappers to it or they add more functionalities to it and provide it as RedHat Operating System.

Similarly, Debain, Alpine, Fedora all of them do the same thing. So basically, they all have their origins common, but they add more wrappers, they add more commands or they add more features and distribute them as new products.

That’s why if you have a program or you have a shell script that is running on Ubuntu most likely it is also works on RedHat because underlying system libraries are common, underlying system packages are common. So, they all work. Like when you write a shell script, it works on most of the linux distributions, but the same thing doesn’t work on Windows because it’s a completely different operating system.

So, which one is popular? Ubuntu is a very popular operating system and if you want to get started, Ubuntu is highly recommended as an operating system because its very easy, it also comes with most of the required system packages and libraries. So, for beginners its hassle-free.

So, this is what we call as Linux Distributions [3].

Now, the next thing is How do we setup Linux [4].

***Thought Provoking:***

**OK Devender!**

* **I understood what is Operating System.**
* **I learnt history of Operating Systems.**
* **I learnt about the structure of Linux.**
* **I know what is Linux Distributions.**
* **And Ubuntu is the popular distribution**

**Now, tell me how can I install Ubuntu on my machine.**

Let’s say you are using Windows machine.

One simple thing that you can do is install WSL (Window Subsystem for Linux). It’s very very simple. Just go to your command prompt or PowerShell in your Windows and run the command ***wsl --install***. So, WSL package is available out of the box on your Windows machine 😊, just go to PowerShell and run above command.

When you run ***wsl --install*** command, it creates Linux -like environment on your Windows machine. It doesn’t create a virtual machine but it creates a Linux-like environment on your Windows machine.

And once this is done, just restart/reboot your machine and go to *command prompt* and run ***wsl*** again. When you do this, you have Ubuntu running and you can access Ubuntu machine on your windows. It’s as simple as that.

Many people don’t use this option and they go for EC2 instances on AWS or they go for virtual machine on Azure. They constantly try to stop the instances, start the instances just to learn Linux.

Trust me, to learn Linux, whatever we are going to do in the 10 sections, you don’t need an EC2 instance. You can just learn that on your personal machine using WSL.

***Thought Provoking:***

**OK Devender!**

**For some reason I cannot run WSL on my machine. Is there any other alternative?**

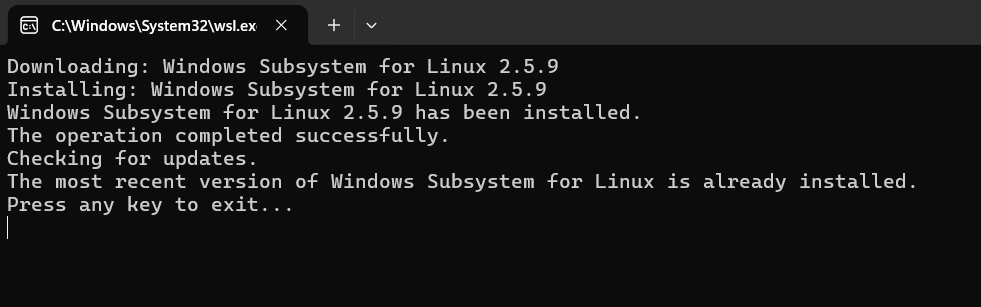
Definitely YES. You can use Docker. If you don’t know what container or what is Docker as a container platform, you don’t have to worry. Just run the command that I’m going to share.



If you go to [04-setup.md](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/blob/main/01-getting-started/04-setup.md) in GitHub repo, which is a sophisticated command that it was designed. It is added with all the properties so that your docker container runs as a Linux machine either on your Windows or Mac OS laptops.

For example, I will copy this [**Docker Command to Run Ubuntu Linux Container in windows host (Persistent & Long-Term)**] as I’m using Windows. (You do require Docker Desktop to be installed on your Windows machine)

On successful installation of Docker Desktop and system restart, you get a popup to update/ install WSL for linux, give confirmation by hitting Enter key. It gets the latest WSL downloaded and installed.



Now Open PowerShell and run below snippet

docker run -dit `

--name ubuntu-container-demo `

--hostname ubuntu-dev `

--restart unless-stopped `

--cpus="2" `

--memory="4g" `

--mount type=bind,source="C:/Users/deven/Downloads/ubuntu-container-demo",target=/data `

-v /var/run/docker.sock:/var/run/docker.sock `

-p 2222:22 `

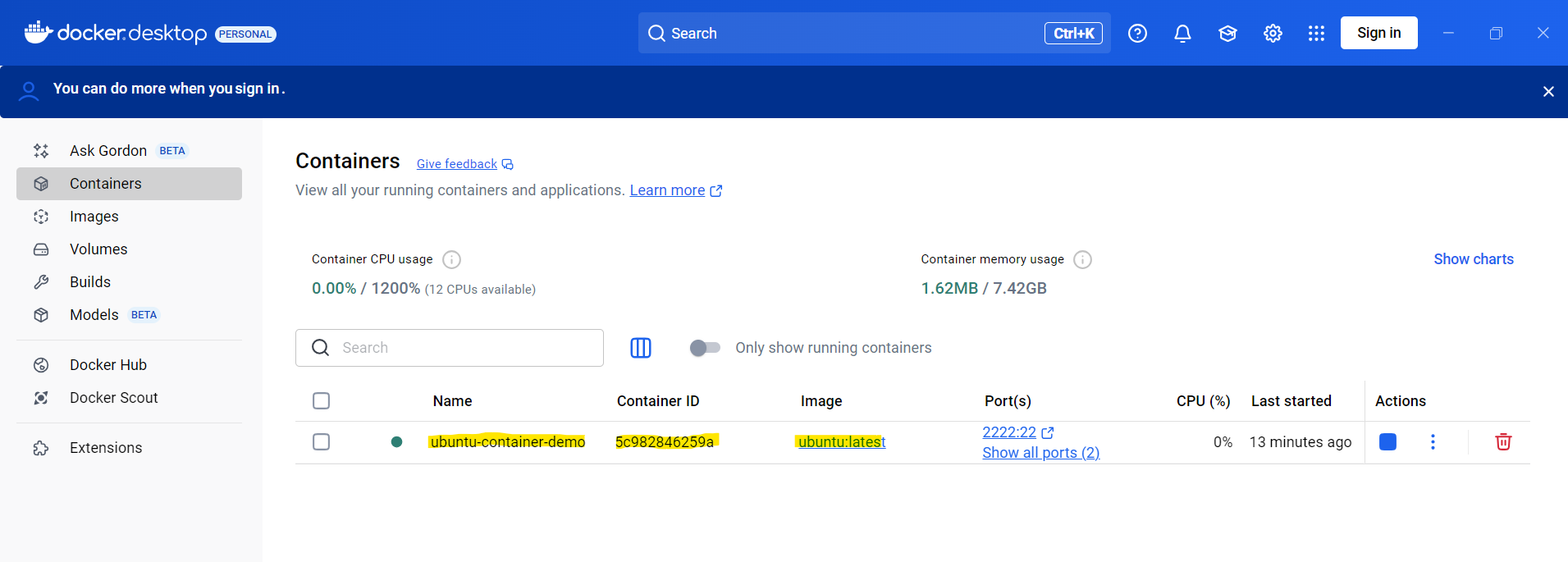
-p 8080:80 `

--env TZ=Asia/Kolkata `

--env LANG=en\_US.UTF-8 `

ubuntu:latest /bin/bash

It will create a docker container on your windows machine

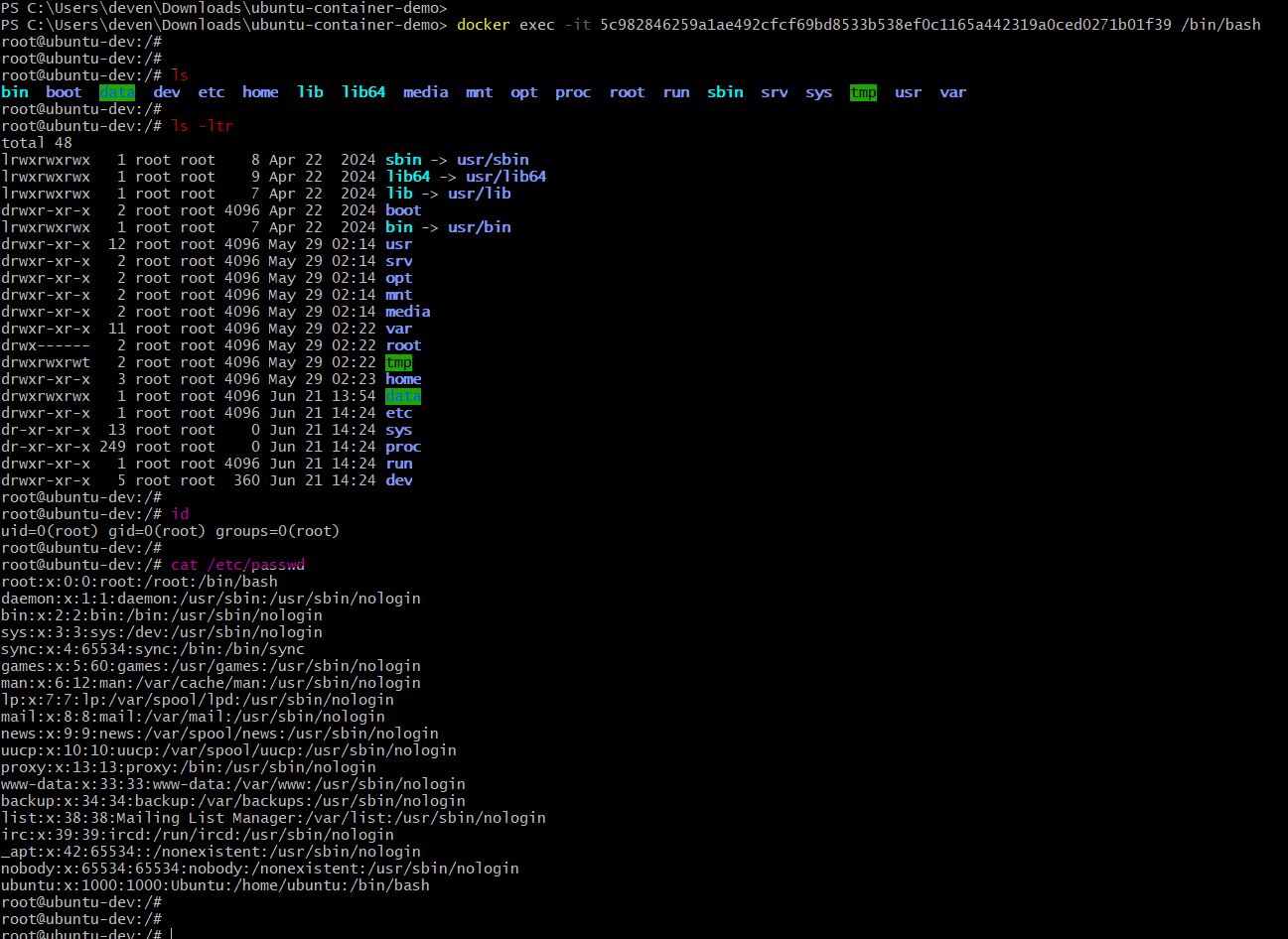


Use below command to navigate or access your container

***docker exec -it <container id> /bin/bash***

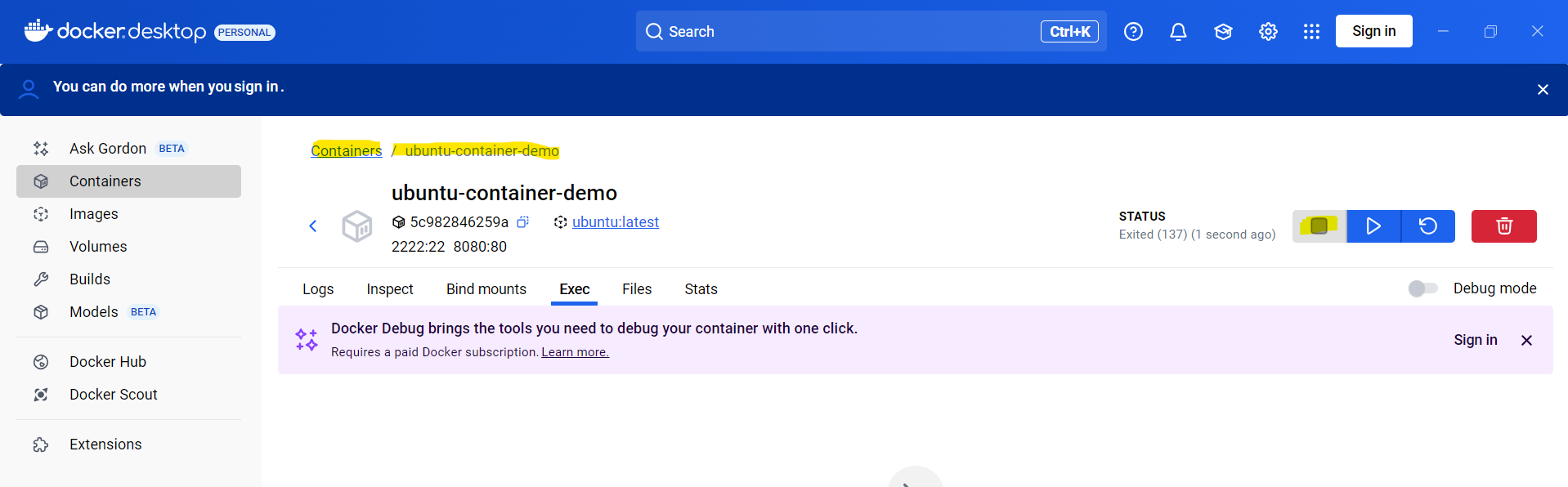
This will take you to inside of the ubuntu container which you have created through docker 😊



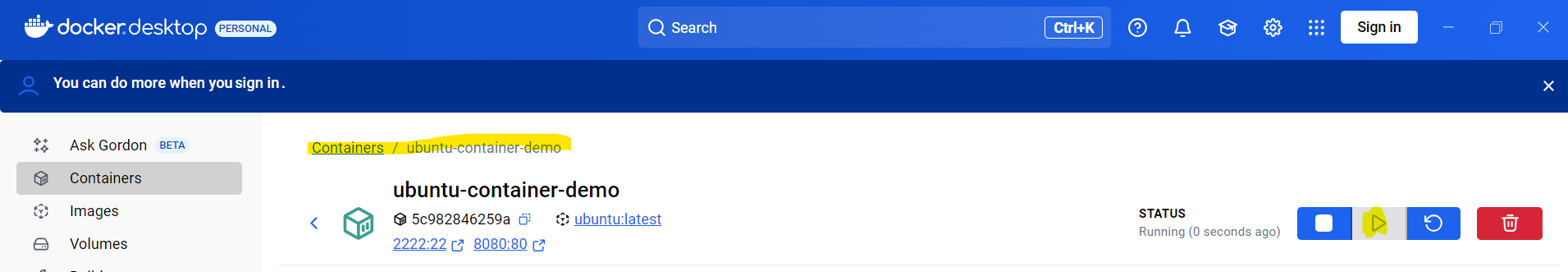


To exit from the docker container from PowerShell, use ***exit*** command

If you want to stop the container, you can do it from Docker Desktop.



If you want to start the container, you can do it from Docker Desktop.

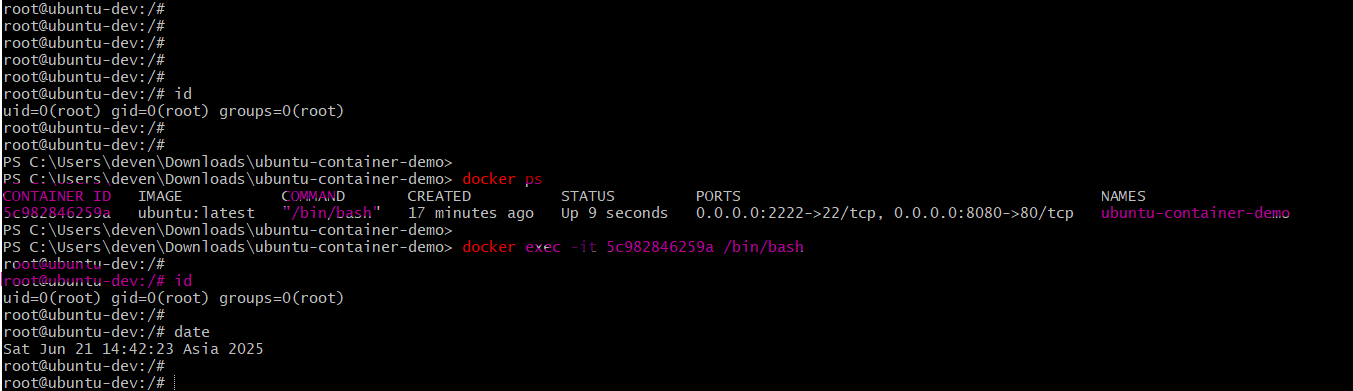


***To stop and start docker container from command line.***

To stop a running Docker container on Linux, use the command ***docker stop <container\_id\_or\_name>.***

The ***<container\_id\_or\_name>*** should be replaced with the actual ID or name of the container you want to stop.

You can find the container ID or name using ***docker ps***



First, ***docker ps -a*** shows all containers (the ones that are running and the stopped ones), so that is the reason you are not seeing your stopped container listed.

Second, you can easily start a stopped container running:

***docker start <container\_name>***

Once the container has been started, you can enter into container and run your commands

***docker exec -it <container\_name> /bin/bash***

Commands

* To see your running and stopped containers.
* To stop the running containers,
* To start your stopped containers.

PS C:\Users\deven\Downloads\ubuntu-container-demo>

PS C:\Users\deven\Downloads\ubuntu-container-demo> date

Sat Jun 21 20:25:10 IST 2025

PS C:\Users\deven\Downloads\ubuntu-container-demo>

PS C:\Users\deven\Downloads\ubuntu-container-demo> **docker ps**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

**5c982846259a ubuntu:latest "/bin/bash" 30 minutes ago Up 2 minutes 0.0.0.0:2222->22/tcp, 0.0.0.0:8080->80/tcp ubuntu-container-demo**

PS C:\Users\deven\Downloads\ubuntu-container-demo>

PS C:\Users\deven\Downloads\ubuntu-container-demo> docker stop ubuntu-container-demo

ubuntu-container-demo

PS C:\Users\deven\Downloads\ubuntu-container-demo>

PS C:\Users\deven\Downloads\ubuntu-container-demo> docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

PS C:\Users\deven\Downloads\ubuntu-container-demo>

PS C:\Users\deven\Downloads\ubuntu-container-demo> **docker ps -a**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

**5c982846259a ubuntu:latest "/bin/bash" 31 minutes ago Exited (137) 13 seconds ago ubuntu-container-demo**

PS C:\Users\deven\Downloads\ubuntu-container-demo>

PS C:\Users\deven\Downloads\ubuntu-container-demo> **docker start ubuntu-container-demo**

**ubuntu-container-demo**

PS C:\Users\deven\Downloads\ubuntu-container-demo>

PS C:\Users\deven\Downloads\ubuntu-container-demo> docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

**5c982846259a ubuntu:latest "/bin/bash" 32 minutes ago Up 4 seconds 0.0.0.0:2222->22/tcp, 0.0.0.0:8080->80/tcp ubuntu-container-demo**

PS C:\Users\deven\Downloads\ubuntu-container-demo>

PS C:\Users\deven\Downloads\ubuntu-container-demo> **docker exec -it ubuntu-container-demo /bin/bash**

root@ubuntu-dev:/#

root@ubuntu-dev:/# id

uid=0(root) gid=0(root) groups=0(root)

root@ubuntu-dev:/#

root@ubuntu-dev:/# date

Sat Jun 21 14:57:13 Asia 2025

root@ubuntu-dev:/#

root@ubuntu-dev:/# exit

exit

PS C:\Users\deven\Downloads\ubuntu-container-demo>

PS C:\Users\deven\Downloads\ubuntu-container-demo> date

Sat Jun 21 20:27:22 IST 2025

PS C:\Users\deven\Downloads\ubuntu-container-demo>

Initially I have created the container and using ***exec*** I went into the container.

Now, you can run any linux command here. You can explore the folders and you can typically do what I’m going to explain in the next sections.

You can perform process management on it, you can write shell scripts in it, you can learn different files and folders, anything that you want to do with a Linux Environment. So, it is as simple as that.

Either go for WSL or use the Docker container to simulate Linux environment on your laptop.

If both of these don’t work, which is extremely rare case, only then go ahead to your cloud platform and create a cloud instance. Once you create a cloud instance, you can access it and start learning Linux. But to avoid costing and make it easy, I would recommend these approaches.

Now, finally, what is Package Manager [5].

So, the last topic for today’s section is understanding Package Managers.

Let’s go back.

So, I have a Linux environment where I’m using Ubuntu. Now, how do I install packages on it?

Let’s say I want to install Python. If you’re using Window’s machine, you can go to the internet, download packages. But when I’m on Linux, how to install packages? Of course, there are commands to install the packages, but what these Linux distributions do?

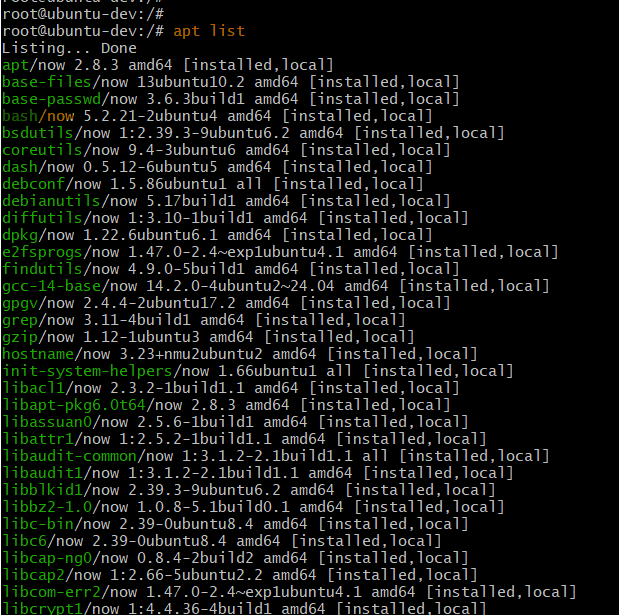
Linux distributions take the open-source code and they add more libraries.

One such thing the Linux distributions do is they add Package Managers?

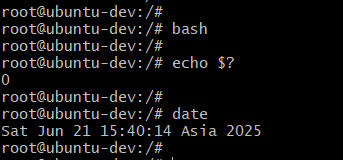
Now what are these package managers? Package Managers help you deploy or install a dependency something like Python, it helps you to upgrade to latest version of the package, delete the version of the package or even maintain the packages.

The best example is Python, Java etc. Using package managers, you can install Python, you can install Java, you can remove Java from your Linux Machine, upgrade Java or continuously maintain the versions of Java.

When you are using Ubuntu, the package manager is ***apt***.So ***apt*** is installed out of the box, so you can just run ***apt list*** it shows list of packages that are available on this machine. So, when you create a Linux machine, already all these packages are installed.

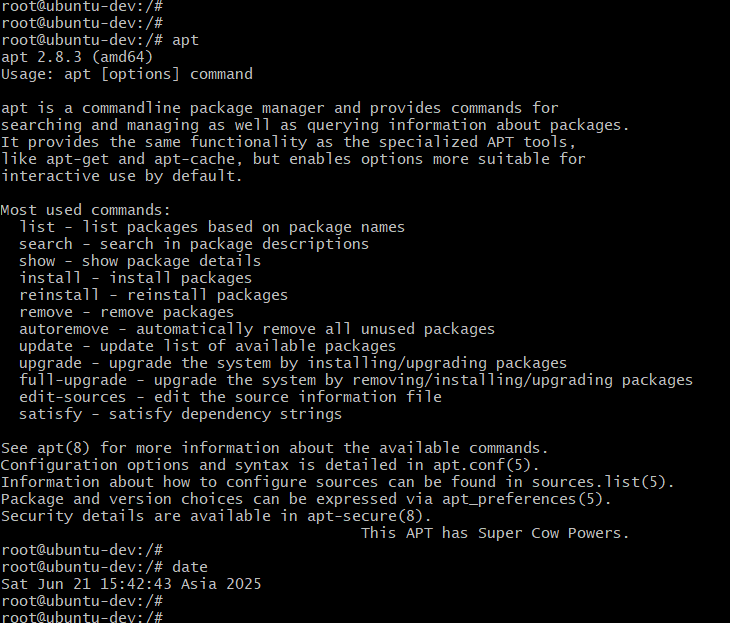


For example, you run Python, if it is already available. Or you run ***bash***



This command ***bash*** got executed because bash is a package that is already installed when you created the Linux machine.

Similarly, when you run ***apt*** again



You can learn what is ***apt list*** command. It lists all the packages.

Similarly, you can run ***apt search*** to search for a particular version of a package, or you can do ***apt install***.

***Thought Provoking:***

**OK Devender!**

**What is *apt install?***

So, when I do ***apt install*** ***python3***

What it does is, it goes to a centralized location. So, these package managers download and cache some dependencies, commonly used dependencies and they store them at a centralized location. For example, Ubuntu stores in **archive.ubuntu.com.**

archive.ubuntu.com: This is a master archive containing all official Ubuntu packages, which are then mirrored on various servers worldwide.

*Ubuntu package manager, APT (Advanced Package Tool), does not store its packages directly on archive.ubuntu.com. While archive.ubuntu.com is a crucial component of the Ubuntu archive, it's a mirror site that serves as a repository for Ubuntu packages, not the sole storage location.*

*APT uses a list of configured repositories (defined in /etc/apt/sources.list and /etc/apt/sources.list.d/) to locate and download packages. These repositories can include archive.ubuntu.com and other mirrors.*

*Here's a more detailed explanation:*

* ***APT (Advanced Package Tool):***

*This is the package management system used by Ubuntu and other Debian-based distributions. It handles the installation, removal, and updating of software packages.*

* ***archive.ubuntu.com:***

*This is a master archive containing all official Ubuntu packages, which are then mirrored on various servers worldwide.*

* ***Repositories:***

*APT relies on configured repositories to locate packages. These repositories are listed in files like /etc/apt/sources.list. These files tell APT where to find the packages it needs.*

* ***Package Lists:***

*When you run apt-get update, APT downloads package lists from the configured repositories and stores them locally in /var/lib/apt/lists.*

* ***Downloading Packages:***

*When you install or upgrade a package using apt-get install or apt-get upgrade, APT downloads the necessary .deb packages from the repositories (including archive.ubuntu.com or its mirrors).*

* ***Local Storage:***

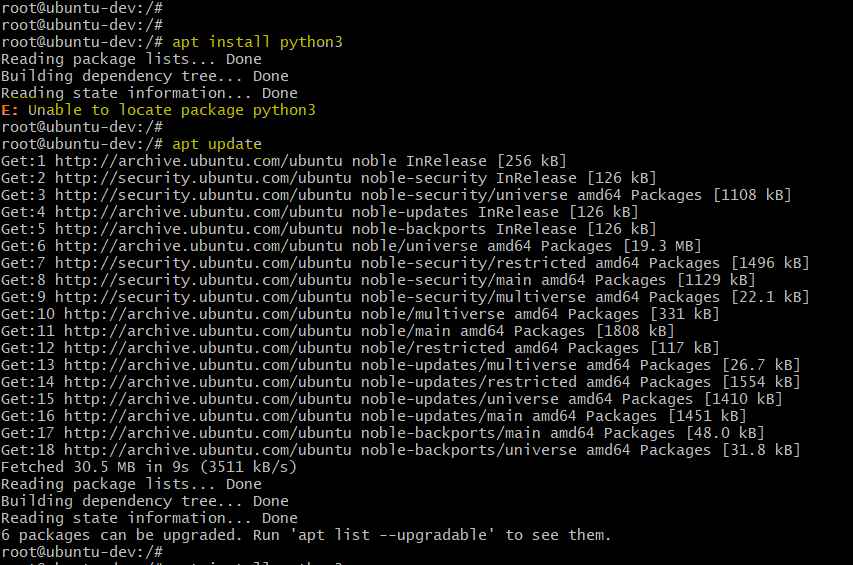
*The downloaded .deb packages are temporarily stored in /var/cache/apt/archives before being installed.*

*So, archive.ubuntu.com is a key part of the Ubuntu package distribution system, but not the only place APT looks for packages, and it's not the final storage location for installed packages on your system.*

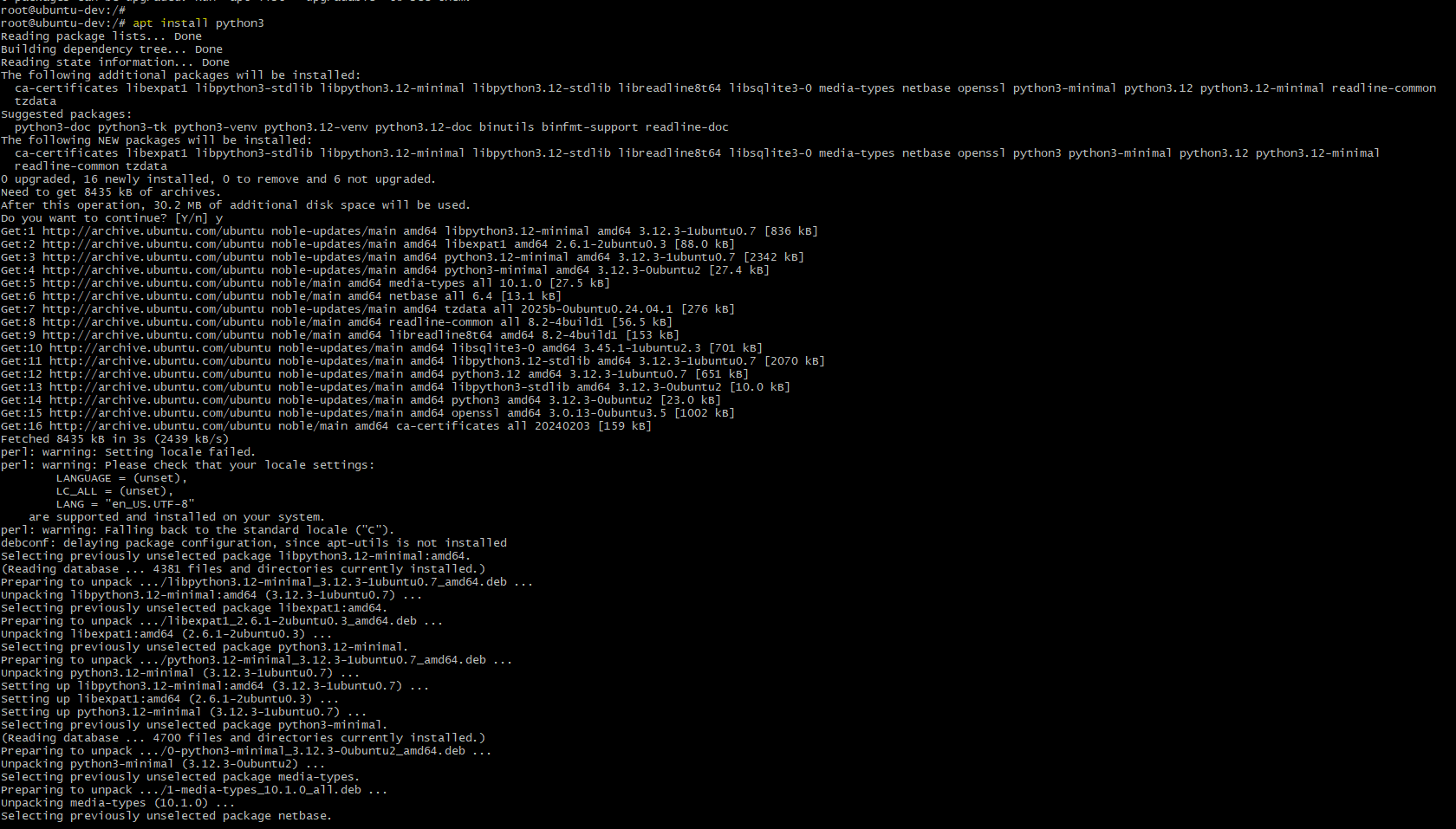
So, whenever you run ***apt install python3***itgoes to that location and downloads ***python3*** from that trusted location.

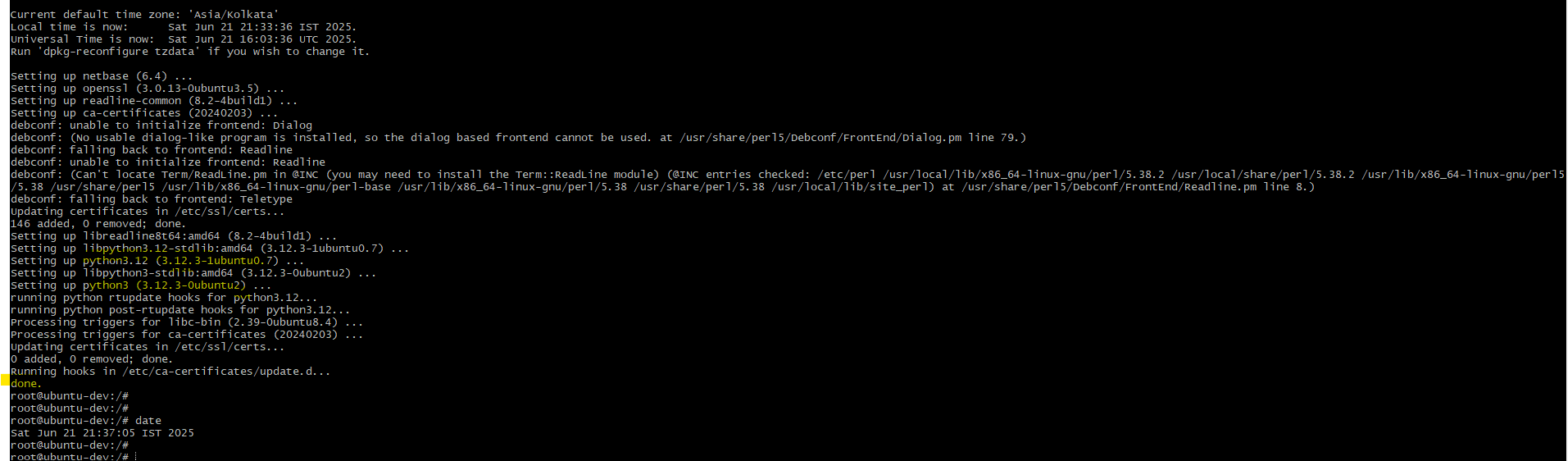
Of course, there are some packages which are not available with the package manager. Then you need to know ***curl*** command or ***curl*** command which you will learn in future sections.

So, that is the use case of ***apt.***



First time when you install, you need to run ***apt update***. It is required because whenever you are installing a Linux virtual machine, this kernel might be packaged sometime back. So, it is important to update all the packages on your machine. Only the you can run ***apt install python3*** then the package gets installed on my machine.





So, this how the Package Managers [5] work.

It also tells us that, it is trying to look for this particular location <http://archive.ubuntu.com/ubuntu> and from this location it is trying to download Python3.

It downloaded and now I have Python3 on my machine. This is the use case of Package Managers [5].

So, what we learned so far?

* We have learned the Fundamentals of Linux, where we learned
  + What is a Linux Kernel
  + What are the responsibilities of it
  + What is Linux Operating System
  + About comparison of Linux with Windows
  + Various Linux distributions
  + The setup of Linux using WSL as well as Docker container
  + Finally, we learned about Package Managers

Linux Setup: This is the important point, going ahead, please try to use WSL or docker container I have shared to use for you. Incase if this doesn’t work for you, use cloud instance as ec2 or azure virtual machine.

So, this is the first section and I tried it to keep it simple 😊.

From the next section, It’s going to be interesting where we are going to learn a lot of new things.