In this section, we will learn ***Let’s explore the concepts as part of Section 3 4 & 5.***

* User Management
* File Management
* Vi shortcuts

Let’s start with User Management in Linux

* + What is User Management in linux?
  + Why is it needed

Before we learn this, let’s go a step back and discuss about our personal laptops.

Many of you might be using Mac OS or Windows laptops.

When you buy them, for the first time, you set up an admin user and password for the admin user.

It’s very unlikely that you go back and create more users on your personal laptop. Although your laptop supports it, but most of the times you don’t create any additional users because it’s your personal machine and you are the one accountable for any action on the machine.

Let’s say a file is created, a file is deleted or a software is installed, you know that you have done it. So, you don’t bother creating another user called non-admin and when you want to perform software installations, use the admin user and when you just want to browse or surf through the internet use the non-admin users. I don’t think many people would do that. Because the accountability on your personal laptop is pretty clear.

Whereas, when you talk about a Linux server, in the previous sections I have explained that 90% of production workloads run on a Linux server. So, it’s most likely when you join a company, you will be working on a Linux server.

Whether you are a developer or DevOps engineer or QA engineer, you would need access to the Linux server because end of the day your applications are running on Linux server.

You might want to look at some files, you want to change some configuration of your application or you want to read the logs of application.

So, imagine there is no **User Management** in Linux, and your company has 100 developers (developer A developer B developer C ….) and all of them are shared with the **root** user and password of the **root** user.

In the last section, we have learned what is **root** user. It’s a super user with all the permissions on the machine.

So, if all of these developers are granted with **root** user and the password of the **root** user and unfortunately a developer deletes a file from Linux server.

Imagine this developer deleted ***/sbin*** folder from Linux server which is System Binaries folder and it has all the important binaries. So, your Linux server is almost corrupt ☹.

If you know who has done this, in future you might restrict access to this user from other Linux servers or you might flag this user as spam.

You can take actions on the user. However, all of them are granted with root user. So, there is no way you know who has done it ☹. So, this is called ***lack of accountability*** and your Linux server is not at all safe when there is no user management.

To solve this problem, what administrators do as part of user management they would create **users** and **groups**.

So, every developer in the organization, every QA engineer in the organization, every DevOps engineer in the organization, anyone who needs access to the server, whether it is 100 or 1000 or any number of servers, individual users are created to those engineers.

For example, there is an engineer with name **Devender Musukula**. So as Linux administrator you would come with a naming convention maybe the first name of the user and the surname of the user. And as a Linux administrator you would create a user with name **DevMusukula.**

This is just to ensure if you just keep the username as **Devender**,tomorrow there might be another **Devender** in the organization. So, to avoid that you would come up with a nomenclature and you would create user’s name which are unique and for each user you would grant permissions depending upon what user do in the organization.

If user is a DevOps engineer, probably you would grant admin access that is the highest privilege. If the user is developer, maybe you would restrict access to folders like **/sbin**.

If the user is a manager, probably you just grant read access to all the files on the server.

We will learn that in detail. We have a section called file permissions where I’ll explain you in detail how to grant permissions to the user.

In this section, we will focus on

* How to create the users
* How to set up passwords for the user
* How to login as an individual user instead of root user.
* And similarly, we will learn about groups as well.

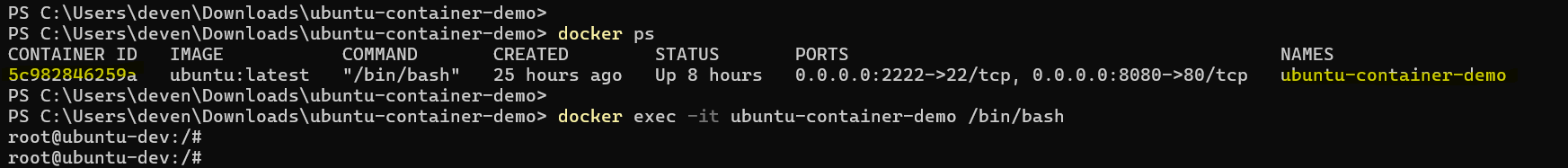
So, let’s go ahead and see how to create users first.

So, for creation users and groups, we will continue with our local Linux Environment.

In the previous section, I have explained you how to create a local Linux environment either using WSL if you are on Windows or go for a Docker environment if you’re on Mac OS. Thos commands are available here [**01-getting-started/04-setup.md**](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/blob/main/01-getting-started/04-setup.md) where it was explained about WSL docker container environment.

So, I will use the docker container environment for teaching you **user management** and **group management** as well.

How to login to that container environment that we already created?



If you haven’t created, create it using the command present at [**01-getting-started/04-setup.md**](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/blob/main/01-getting-started/04-setup.md) and if you have already created, just search for the container using **docker ps** command and look for the container with name **ubuntu-container-demo**, go to the container Id. You can select either container id or container name and execute command **docker exec -it <docker container id /container name>** followed by **/bin/bash.**

***Thought Provoking:***

**Hey Devender!**

* **I don’t want to use any of these.**

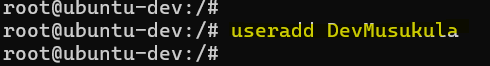
No problem 😊!

You can just go to your favorite cloud provider and create an instance. If you don’t know how to create an instance follow [Step 1: Launch an instance](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html#ec2-launch-instance)

Once you run the command ***docker exec -it ubuntu-container-demo /bin/bash*** you are inside the Docker Linux container environment.

So, how to create a user? Let’s start with that.

To create a user in Linux, you will use this command **useradd** followed by name of the user. So, let’s use the same nomenclature **useradd DevMusukula**

****

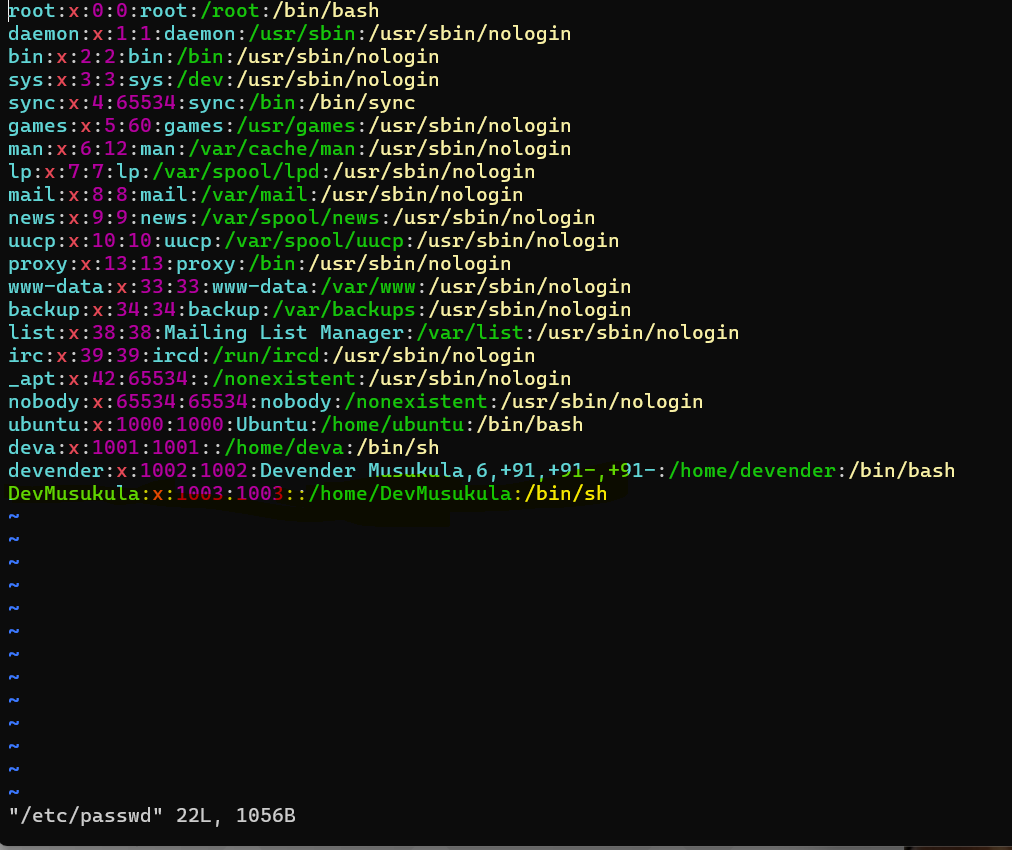
And this will create a user called **DevMusukula** on my Linux Machine.

***Thought Provoking:***

**Hey Devender!**

* **How can I check it if the user DevMusukula is created successfully?**

So, for that there is file in Linux which is **/etc/passwd**. So, **/etc** is a system configuration folder, it’s just like settings, within that there is a file called **passwd**.



If you open **/etc/passwd** file you will find a user **DevMusukula** is created. So, if there is a entry in this particular file **/etc/passwd** file that means a user is created with that particular username.

If in your case ***vim*** doesn’t exist, just run **apt install vim** or you can use **cat /etc/passwd** to directly print the contents of the file on your terminal.

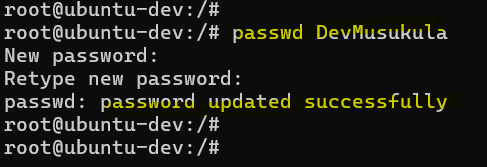
However, a user without password is of no use. So, how to create password for the user?

For that, there is a command called **passwd** followed by **name** of your user.

*passwd <username>*

You need to create a password i.e. administrator sets up the password and shares this information securely. In companies there are a lot of internal tools to share such user credentials.

So, once the administrator creates this username and password, through that internal security tools, administrator will send that to the developer, DevOps engineer, QA engineer whoever it is.



So, now my user is created.

***Thought Provoking:***

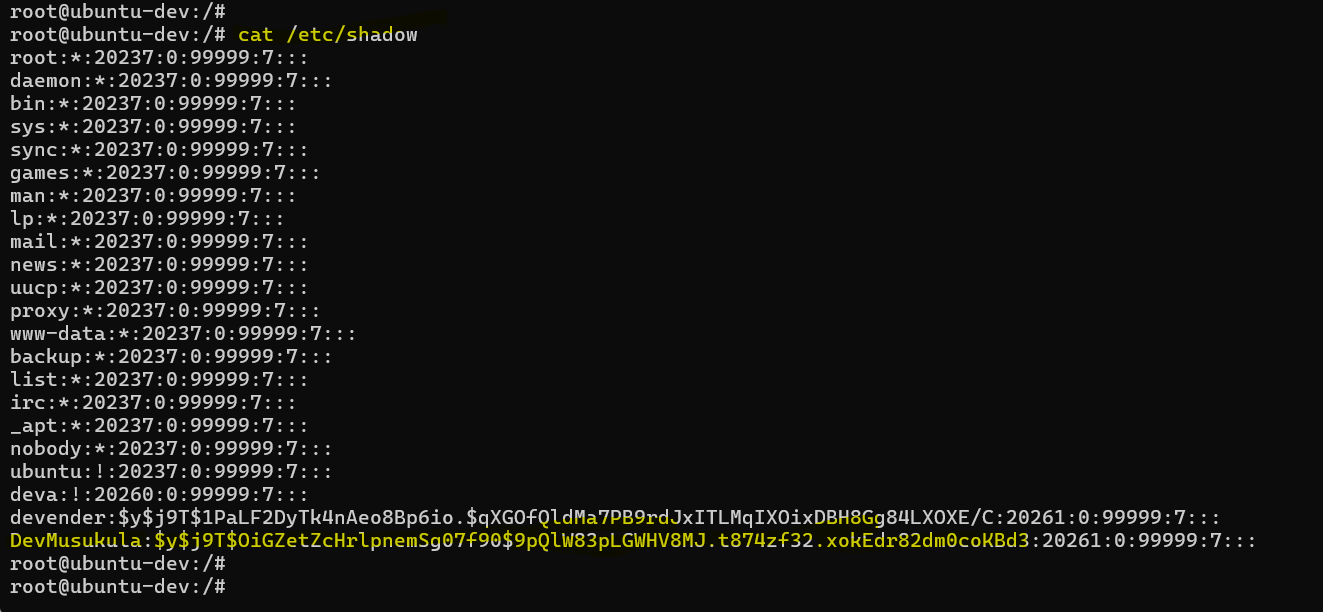
**Hey Devender!**

* **How can I check it if the password is created?**
* **You showed how the user is created. Now I want to know to see if the password is created or not.**

You can definitely check that 😊.

There is a file just do **cat** to print output of the file to the terminal.

***cat /etc/shadow***



So, for **DevMusukula** this is the encrypted password.

***Thought Provoking:***

**Hey Devender!**

* **Can I decrypt this password?**

It is an important as well.

So, if you have access to a Linux server, can you decrypt passwords of users on the Linux machine or this can be asked in other ways as well.

As an administrator you shared username and password to a developer and after 1 month developer forgot the password.

So, can you restore the old password of the developer? In both of the cases, answer is ***NO***.

Although you have encrypted password here, this is very highly encrypted. So, it is as part of this encryption SHA 256 I believe, so, there is a hashing done and this is a one-way encryption. So, decryption is not possible.

So, even though you have access to a Linux server, you can go to this file **/etc/shadow** and look at the encrypted password but you cannot decrypt it.

In a nutshell, if somebody forgets the password, there is no way to restore the old password. There are some tools on the internet, which promises to do it. However, they are not at all safe. They decrypt basic passwords but they are not allowed to use in the organization.

So, we have learned

* useradd to add a user
* passwd to create password for the user
* cat /etc/passwd file which has all the user information.
* /etc/shadow file which has encrypted password

Now we have learned all of these things, Imagine I have to delete a user.

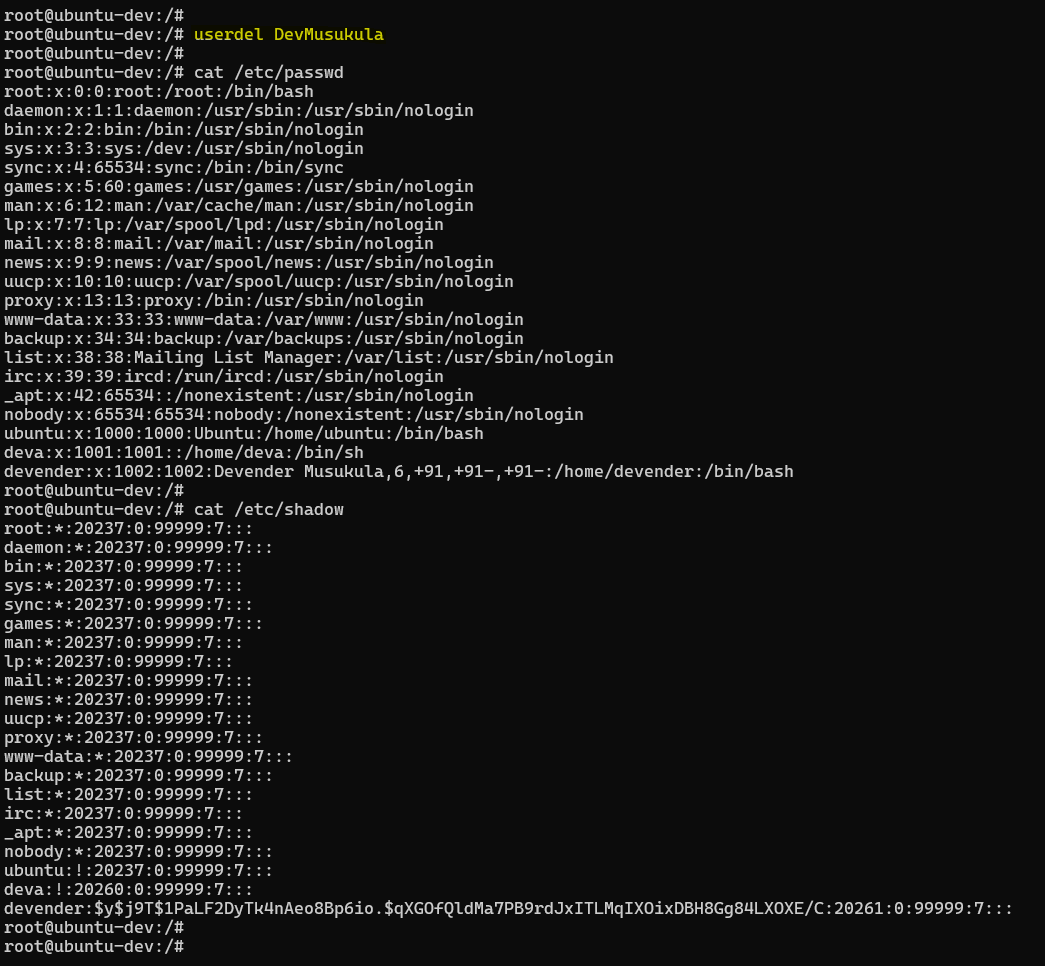
I have created a user, but after a while user left the organization or is leaving the organization. How can I delete user for the organization or user from the organization?

Very Simple 😊! All that you need is **userdel** followed by the username

***userdel <username>***

******

That’s it! User **DevMusukula** is now deleted.



Now, if I search for **/etc/passwd** you will not find the user called **DevMusukula.** It’s as simple as that 😊!

If a user is leaving the organization just do that.

***Thought Provoking:***

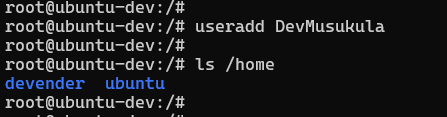
**Hey Devender!**

* **You have created a user with useradd but details of the user are not saved?**

Sometimes, you want to take the details of the user.

For example,

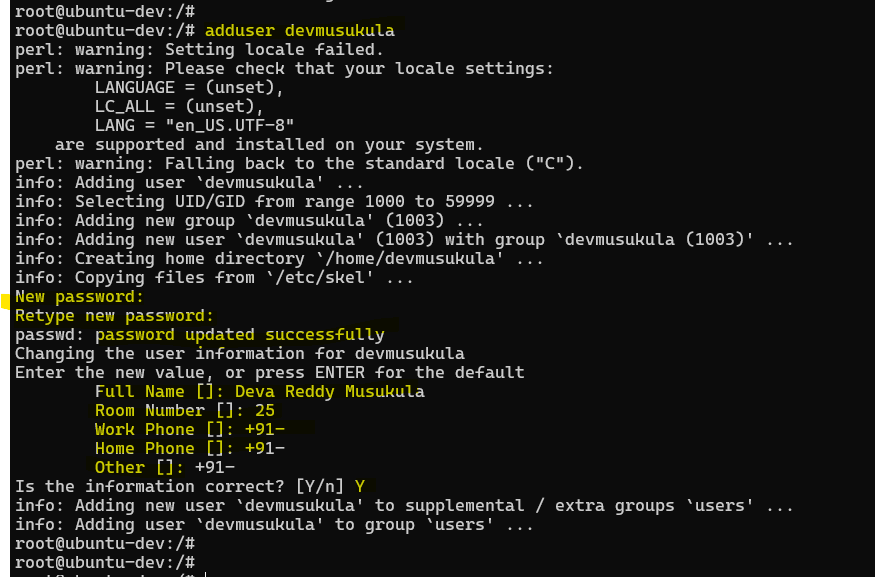
* + What is the Full Name of the user
  + To which group this user belongs to
  + And you also want to create a home directory for the user



When I did **useradd** **DevMusukula,** although user is created, but if you just look at **ls /home** there is folder for **ubuntu** and there are folders for other users **devender** I have created in the past, but for **DevMusukula** there is no home directory. Why?

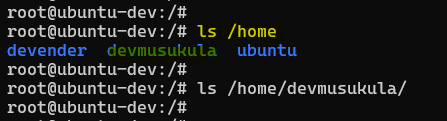
So, **useradd** is a simple command to create user. It doesn’t create home directory for the user or it doesn’t save user details.

To do that, this similar command (useradd), instead of **useradd** you should do use **adduser** followed by **DevMusukula**. *adduser <username>*



This time it started doing some things and asked password which we have provided. Full Name of the user I would say **Deva Reddy Musukula**. Then it says Room Number, Work Phone, Home Phone, Other details can be skipped. We will say all the information is correct. And then it said a user is created and also home directory is created for the user.

Let’s verify 😊.



User **devmusukula** now has a home directory as well.

***Thought Provoking:***

**Hey Devender!**

* **Now all this is fine, you said user management is important for accountability and keeping Linux environment secure. Can you prove this?**

Sure 😊!

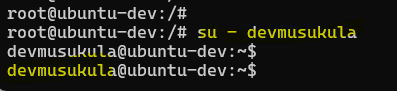
So, what will we do?

We are in the **root** user, let’s switch to **devmusukula** user and try to delete ***/sbin***.

Let me try to do it. I’m ready to corrupt my Linux environment. If what is said is wrong, my Linux environment will be completely corrupt. If user management in Linux environment doesn’t work, I’m going to mess up my Linux environment.

To switch to user **devmusukula** there is a command called **su - <username>**

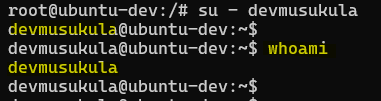
***su – devmusukula***



So, you can see, from **root@ubuntu-dev** I went to **devmusukula@ubuntu-dev.**

Previously it was root@ubuntu-dev and now it is devmusukula@ubuntu-dev.

I can also check that using another command called **whoami**



***Thought Provoking:***

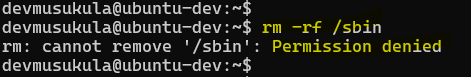
**Hey Devender! You are using so many commands ☹**

Don’t get confused with the commands 😊.

It’s Okay!

With practice you will also learn it and I’m intentionally trying to put these commands in place in order so that you will never forget them. I’m randomly not using switch user (su); I’m randomly not using **whoami.** This is a very simple command I could have taught on day one as well. But the best place for teaching this command is right now. That’s why I explained it at this point.

Going back now I’m going to risk it 😊. ***rm-rf /sbin***



You can see it says *Permission denied*

That is the power of **User Management** in Linux.

When you create users, by default they will not have permissions like the **root** user.

***Thought Provoking:***

**Hey Devender!**

* **How can I grant permissions to devmusukula?**

Don’t worry 😊! You will learn that as part of [06-file-permissions](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/tree/main/06-file-permissions) in the next sections.

For now, let’s not jump and dive to that point 😊.

If I go back and run the same command **rm -rf /sbin** as **root** user, all the folder will be deleted. If somebody wants to risk it, they can 😊.

Till now, we have learned a lot of things. I want to stress on two things,

* What is the difference between **useradd** and **adduser**?
  + Both of these commands do the same thing. However, **adduser** creates home directory for **devmusukula**. It (adduser) also takes more user information.
  + Whereas **useradd** is a quick way of creating the users where it doesn’t prompt for details, it doesn’t ask user full details, or it doesn’t create the home directory.

***Thought Provoking:***

**Hey Devender!**

* **Then what is the use of useradd because adduser looks to be more powerful?**

Very Simple 😊!

The **useradd** is useful when you are writing scripts. So, when you are writing scripts you don’t want the tool or the command t prompt inputs or to ask inputs from you.

Because this **adduser** becomes an interactive way. Whereas, when you are writing a shell script or python script you want a command that just creates a user and goes to the next instruction of the shell script.

So, only in such cases, you will go with **useradd.** When you write shell scripts to create user, to create automation through the user, you will realize importance of **useradd.**

Then we learned another important stuff

* Can you restore the password of a Linux user?
  + The answer is **NO**
  + If a Linux user forgets the password and the password is not stored either in a file or in LDAP or maybe in SSO of the organization, there is no way to restore the password. Although you can get encrypted password from **/etc/shadow** file.

I would highly recommend at this point of **user management** like once we complete the section, just go back and thoroughly read the [README.md](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/blob/main/03-user-management/README.md) file from [03-user-management](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/tree/main/03-user-management).

Because in [README.md](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/blob/main/03-user-management/README.md) file, I have few commands more which I not explained during the section overview. Because its quite obvious I might have missed few things 😊. But if you go through this document [README.md](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/blob/main/03-user-management/README.md), you will learn those important commands as well.

For example, when you create password for a user. Let’s say as an organization you have a compliance that every 90 days password have to be changed so that your passwords are not leaked or somebody writing passwords on their workstation or on a whiteboard, after 9- days even if the person forgets to swipe it or somebody knows the password, it will be changed. So, for that you can use this command ***chage -M 90 username***.

Once you create password for the user, run *chage -M 90 username* and then password will be changed every 90 days or it will ask you to change.

Then *Lock a user account*

*passwd -l username*

For some reason as an administrator, you found one of the user’s to be hacky or one of the users is performing some spam activities on Linux server. So, what you can do is use this *passwd -l username* command to basically lock the account, and once you talk to the user and once you feel confident about the user you can just unlock the user account using this command *passwd -u username*

So, we learned about deleting the users. Now let’s go to groups. This was pretty much about users 😊.

Now, as part of **user management** let’s learn about groups.

***Thought Provoking:***

**Hey Devender!**

* **I understood the significance of users**
* **We are creating users**
* **We are creating passwords to the users**
* **We are granting permissions to users to certain files**
* **Why do we need groups?**

Very Simple 😊!

So, the point is that is you have 5 users in the organization, you can manage it through users. Imagine you have 10 users, even then its fine. But let’s say you have 500 users on the Linux server out of which 250 are developers, 200 are QA engineers, 30 are DevOps engineers and 10 are from the management. Just imagine this.

And one fine day, you are a Linux administrator and you are asked to change a particular file permission for 250 developers.

So, right now 250 developers have access to ***/sbin*** folder and you are asked to remove their access from ***/sbin*** folder.

Imagine going to each and every user, going to file and trying to update permissions of 250 developers isa very tedious activity.

Now, what if the users are 1500 instead of 500? Its going to be even more tedious activity.

So, what you do instead is basically you create the groups.

So, you will create group called **dev QA devops** followed by **management.**

And what you will simply do as a one-time activity you will put all these 250 developers in the **dev** group.

Now, if you get such request, all the developers should be revoked access to ***/sbin***folder. Instead of doing it at user level, you will just go to the group and you will modify the permissions of the group. When you do this, directly or indirectly all the developers within the **dev** group will be modified the access. That’s it 😊!

So, groups are basically used for the same reason. At user level as the number of users grow, managing permissions is going to be difficult. Changing permissions, deleting a particular user in a group is going to be very difficult.

But if you create groups, you can directly do that at the group level. Its onetime activity to put users in **dev** group.

***Thought Provoking:***

**Hey Devender!**

* **Can you show this to me?**

Why not? 😊!

Let’s go back to our Linux environment.

First let’s learn how to create a group.

Just like **useradd** its *groupadd*followed by groupname.

***groupadd devops***

***Thought Provoking:***

**Hey Devender!**

* **Is the group created?**

Let’s check 😊



Use *cat /etc/group* you will find a new group called as ***devops.***

So, when I do ***cat /etc/group*** it creates a new group called as **root.**

***Thought Provoking:***

**Hey Devender!**

* **Why there are other groups?**

Let’s check 😊

I have created some groups previously and you have some groups there are some default groups as well.

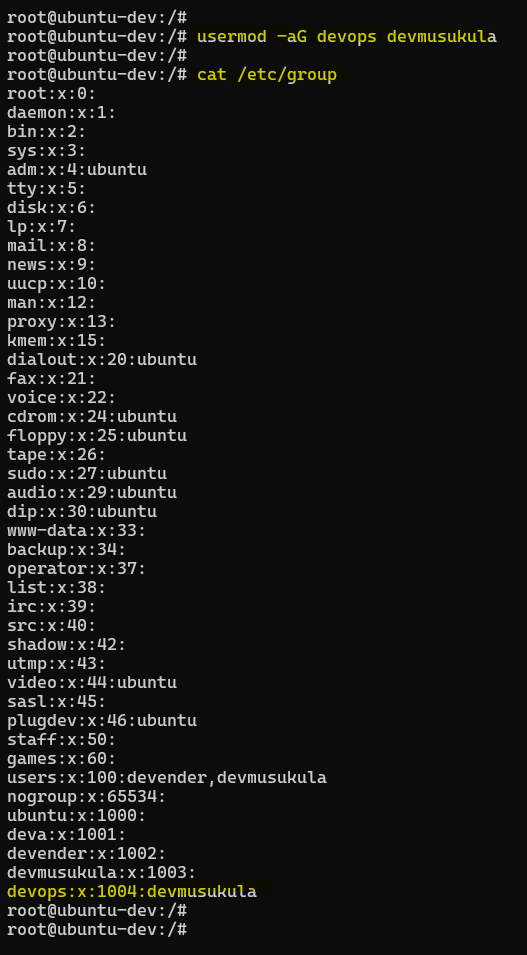
There are some groups also, I will talk about it.

But for now, this is the group **devops** that I have created.

So, it says **devops** is a group and right now there is no user in the group. Let me try to add one.

So, how to add a user to group? For that you have a command called usermod -aG <name of the group> <username>. That’s it 😊!

***usermod -aG devops devmusukula***

******

Now, I go back to the file **/etc/group** and print its contents, you will see *devops:x:1004:devmusukula*

devops à group name

1004 à group id

devmusukula à username

So, that means **devmusukula** is a user which is part of **devops** group.

Similarly, I can add 100 users, 1000 users as part of a single group. When I change the permissions of this particular group, the permissions of all the user’s part of the group will be changed.

So, we learned

* How to create group using ***groupadd***
* Then we learned about **cat /etc/group** file to list out the groups.
* And then we learned how to add user to a group using **usermod -aG <groupname> <username>.**

So, this is about **user management** and **group management**.

But how does this entire thing work in real-time?

For example, when I join a company, I will not be asked to login to a Linux container environment. I will be asked to login to a real Linux server. Or on cloud platforms like AWS or Azure companies have something called virtual machines which is basically a machine on the physical Linux server and I should be logged in to the Virtual Machine.

***Thought Provoking:***

**Hey Devender!**

* **How can I do it?**

Very Simple 😊!

Whatever concept you have learned till now are perfectly fine. Only thing that you need to know additionally is when you join a company obviously, they have machines like this for example, this EC2 instance on AWS isa Linux virtual environment. So, your manager or DevOps engineer, server admin whoever is responsible for it, they will provide you the EC2 instance ip address and they will provide you a username and password. Then you need to login to the EC2 server and typically work on it just like how you are working on docker container Linux environment.

So, what your administrator will do is, they have this option called C**onnect** here and they will connect (this doesn’t ask password or anything). Just like how we were creating on the Linux root environment or docker environment, by switching to **root (**using **sudo su –** to switch from ubuntu to root user. I will tell you what is **sudo**, for now ignore it**)** they will create a user called **devmusukula** through ***useradd devmusukula*** command and then ***passwd devmusukula***.

Now, what the Linux administrator will do? Linux administrator or system admin will send you the ipaddress, username and password in an encrypted way through an internal tool.

Now, to connect to this EC2 Linux environment, what you need is ***ssh client***.

So, we will learn a new topic at this point. And this topic is ***ssh***. Very very simple topic in Linux 😊.

So, every Linux environment, so whenever you create a Linux server, out of the box on the Linux server, there is a process that is running which is ***sshd.*** And this ***sshd*** process will allow you to login to the Linux server. How?

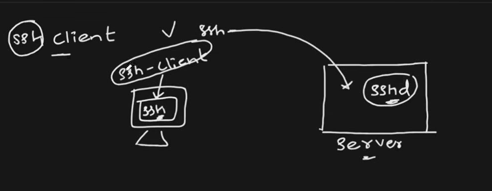
On your personal laptop or on your office laptop because you are working for a company, you have an office laptop environment and on this you install something called as ***ssh*** ***client.***

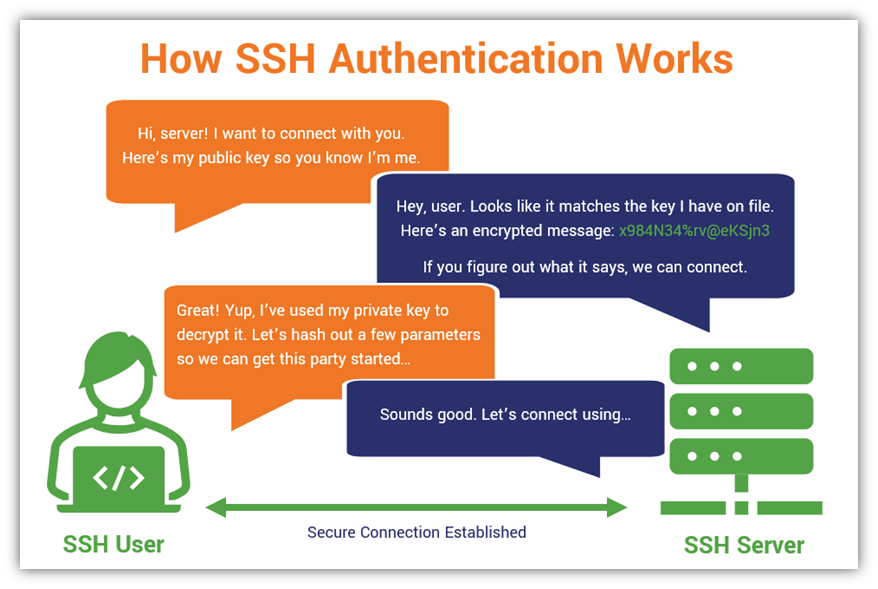
So, when you install ***ssh client*** on your office machine, this SSH client comes with a package called **ssh**.

You might be thinking it is very complicated. **No,** it isa one single step and one single command. Once you learn it, you will feel that “Oh its super simple!”.

But it’s worth understanding the concept. That’s why I’m explain it.

So, by default, a Linux server has a process running called SSHD. On your personal laptop, you will install *ssh-cleint* which will install **ssh** on your machine and you will take this **ssh-client** run a single command with which you will connect to the Linux server





***Thought Provoking:***

**Hey Devender!**

* **What are these SSH clients?**

Popular one if you’re on Windows is ***Git Bash***. Or you can use WSL as well. But Git Bash is very powerful.

If you are on Mac, you can use terminal of Mac as you get it out of the box on Mac. If you don’t get it, go to internet and search for ***iTerm 2*.**

If you’re on Windows search for **Git Bash** and just download it and install it. And you will find a terminal same as that of Linux environment created on docker container.

Press Ctrl+D to come out of the docker container Linux environment. To login to this EC2 Linux machine, on the ssh-client run this command **ssh** followed by the user **devmusukula** that was given to me and @ the ipaddress of the instance.

***ssh*** [***devmusukula@3.83.248.52***](mailto:devmusukula@3.83.248.52)

Hit enter, and it asks you for the password. Just provide the password that is shared by the administrator. And now you are inside the EC2 instance Linux environment.

You can check that if you execute **whoami** command and it will say you’re devmusukula and if you just run the command **uname -a** it will provide that this is the Linux instance.

So, it is very simple to connect to any Linux server once we have the user credentials.

Press Ctrl+D to come out of it. Use the command ***ssh*** [***devmusukula@3.83.248.52***](mailto:devmusukula@3.83.248.52)

again, to login to EC2 Linux instance.

***Thought Provoking:***

**Hey Devender!**

* **But when I try to do it on one of the virtual machines or the cloud providers, password was disabled or password access was disabled?**

Yeah, you are right 😊.

So, what these cloud providers do out of the box. Because some companies don’t manage the passwords well. So, passwords are something that gets leaked because people write in on the s=desk, people write it on the wall, people write it on a paper or just store that somewhere and they just get leaked.

So, to avoid such things, what cloud providers do is.

If you go to the EC2 Linux instance, there is a file on the cloud provider. The location of the file is ***/etc/ssh/sshd\_config.d/60-cloudimg-settings.conf***. Within this file they will set up **PasswordAuthentication** as **no** in this file.

By default, the cloud providers set Password Authentication as **no.** If you are working in your company, your administrator can enable it, disable it and it totally depends on your organizational compliance standard.

***Thought Provoking:***

**Hey Devender!**

* **How did you do it?**

I went to this file ***/etc/ssh/sshd\_config.d/60-cloudimg-settings.conf*** and I have changed ***PasswordAuthentication no*** to ***PasswordAuthentication yes*** and ran this command called sudo *systemctl restart ssh* If you are on Ubuntu user you need sudo and if you are on root, you do not need it just run systemctl restart ssh. That’s it, password authentication will be enabled.

***Thought Provoking:***

**Hey Devender!**

* **If password authentication is disabled, how can I login?**

If password authentication is disabled, then you have something called as ***pem*** file.

Will cover about ***pem*** file in future sections. Right now, we are learning about user management so it is not required.

This was about [03-user-management](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/tree/main/03-user-management). At this point you can have a break and let’s focus on next topics which are super super simple 😊.

Unlike your **user management** next topics are going to be very light.

Go back to the Linux container environment. Now, what are we going to learn?

We are going to learn [04-file-management](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/tree/main/04-file-management). Most simple activity on your Linux environment 😊.

First thing,

* How to list the files?
  + Just go to the **root** user and type **ls.**
* How to go to one of these directories shown in **ls** command output?
  + For that the command is **cd tmp**
  + Again, I can do **ls**, there is a user called **devender.** How do I know this is a folder? looking at the color **blue.**
  + **cd devender** and do **ls** and see if there are any file/folders.



***Thought Provoking:***

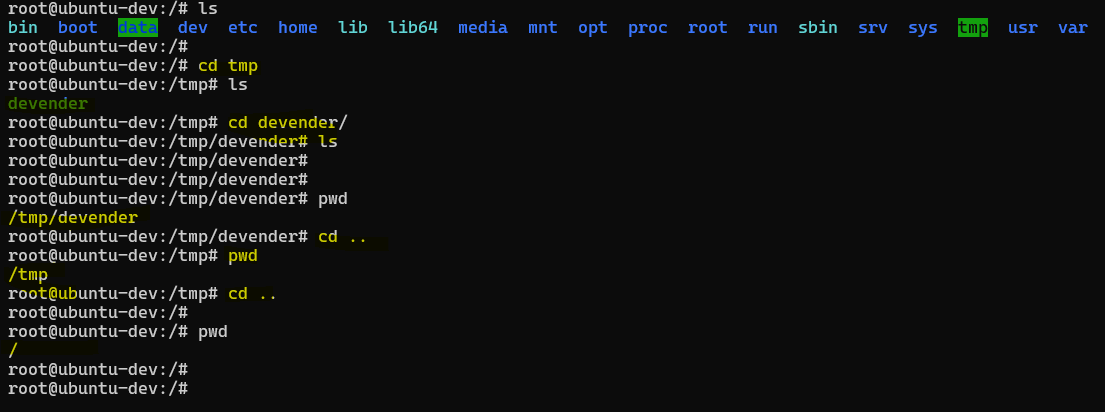
**Hey Devender!**

* **How can I come back?**

I went to **/tmp/devender** folder to comeback you can use **cd** command but just write **..**

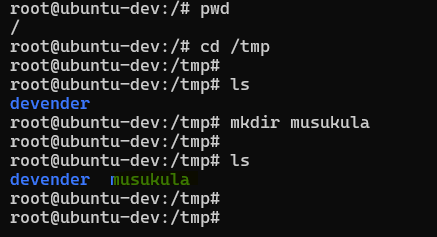
Now, you will be in **/tmp** folder.

**cd ..**

****

You are at the root of the file system.

The command ***pwd*** tells you the current path or the directory you are in.



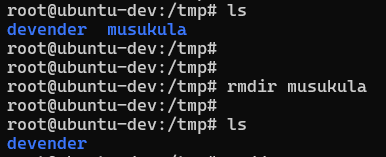
Let’s stay in **/tmp** and create a folder in it. To create a new folder (called as directory in Linux) the command is ***mkdir***. Let me create directory called **musukula**. And if I do **ls,** I will find a folder named **musukula**. As simple as it is 😊.

***Thought Provoking:***

**Hey Devender!**

* **Now, I learned how to list files.**
* **How to check the working directory**
* **How to create directories**
* **How to go back in directories**
* **Now, I want to delete the directory musukula which is just created. How can I do it?**

Its ***rmdir <name of the directory>*** i.e. ***rmdir musukula*** that’s it.



If I do **ls** now, you will not find the directory **musukula.**

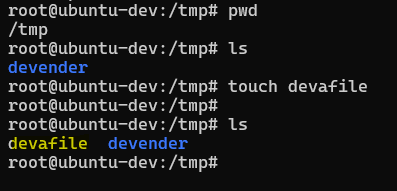
***Thought Provoking:***

**Hey Devender!**

* **Instead of directory, I want to delete a file.**

So, to delete a file, first let’s learn how to create a file.

There is a command called **touch.**

****

***touch devafile*** will actually create a file.

**File and Directory Management**

1. **ls** – Lists files and directories in the current location.
2. **cd /path/to/directory** – Changes the working directory.
3. **pwd** – Prints the current working directory.
4. **mkdir <new\_folder**> – Creates a new directory.
5. **rmdir <empty\_folder>** – Removes an empty directory.
6. **rm file.txt** – Deletes a file.
7. **rm -r folder** – Deletes a folder and its contents.
8. **cp file1.txt file2.txt** – Copies a file.
9. **cp -r dir1 dir2** – Copies a directory recursively.
10. **mv old\_name new\_name** – Moves or renames a file or directory.

Before learning about File Viewing and Editing, let’s actually learn how to write to a file using [05-vi-shortcuts](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/tree/main/05-vi-shortcuts).

**File Viewing and Editing**

1. **cat file.txt** – Displays file content.
2. **tac file.txt** – Displays file content in reverse order.
3. **less file.txt** – Opens a file for viewing with scrolling support.
4. **more file.txt** – Similar to less, but only moves forward.
5. **head -n 10 file.txt** – Displays the first 10 lines of a file.
6. **tail -n 10 file.txt** – Displays the last 10 lines of a file.
7. **nano file.txt** – Opens a simple text editor.
8. **vi file.txt** – Opens a powerful text editor.
9. **echo 'Hello' > file.txt** – Writes text to a file, overwriting existing content.
10. **echo 'Hello' >> file.txt** – Appends text to a file without overwriting.

So this is about [03-user-management](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/tree/main/03-user-management) [04-file-management](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/tree/main/04-file-management) and [05-vi-shortcuts](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/tree/main/05-vi-shortcuts).

It might be a lengthy section but we have covered a lot of interesting things.

I hope you found it well and I would highly recommend to go back to the documentation.

I spent a lot of time adding this documentation. So please go through it.