**DEVOPS NOTES -2**

**LINUX & WINDOWS**

Linux is a one of the Popular Operating system of UNIX OS, Its Open Source as its source code is Easily Available on Internet.

Its Popular Bcoz of its Low Cost, Scalability, Performance, Light Weight, Stability

Linux is used in Server, mainframes, Supercomputers, Embedded System like Smart Phones, Routers, TV, Gaming Console its used.

**WINDOWS OS:** Its Graphical user Interface (GUI) OS and it’s a System S/W. Its Developed and Published by Microsoft and it was released on 20th Nov 1985

In Windows User interact with System with Windows, Icon, Minus and Pointers (WIMP)

**File System**:

* **NTFS**: Windows primarily uses the NTFS (New Technology File System) for file storage, which supports large files, security features, and file compression.

**Security Features**:

* **User Account Control (UAC)**: Helps prevent unauthorized changes to the operating system by requiring administrative approval for certain actions.
* **Windows Defender**: Built-in antivirus and anti-malware protection to safeguard against security threats.
* **BitLocker**: Provides full disk encryption to protect data on the system

**Summary**

Windows OS is a versatile and widely used operating system with a rich set of features designed for both personal and professional use. It is known for its user-friendly interface, broad software compatibility, and regular updates, making it a reliable choice for many types of users.

**Task Scheduler in Windows**:

Task Scheduler Is a Powerful Tool which allows to automate the task by Scheduling Scripting, Programming or Command to run at specific Time or in respond to any event.

**Steps to Configure Task Scheduler:**

1. **Accessing Task Scheduler:**
   * Press **Windows + R** to open the Run dialog.
   * Type **taskschd.msc** and press Enter.
   * Alternatively, you can search for "Task Scheduler" in the Start Menu.

Here **taskschd** is a shortform of **task scheduler** which allow to **create, manage and monitor the task that the computer will execute automatically at specified time or an event**

And **msc** is an extension of **MICROSIOFT MANAGMNET CONTROL** which is a main frame **which provided GUI and Programming Environment**.

1. **Create a New Task**:

* **Navigate to Task Scheduler Library**: In the left-hand pane, click on "Task Scheduler Library."
* **Create Task**:
  + In the right-hand pane, click on "Create Task" to open the "Create Task" window.
  + Alternatively, for a simpler task, you can choose "Create Basic Task," which provides a guided wizard.

1. **General Settings**:

* **Name and Description**:
  + Enter a name for your task in the "Name" field. This should be something descriptive, like "Backup Task."
  + Optionally, add a description for more context.
* **User Account**:
  + Choose the user account under which the task will run.
  + Select "Run only when the user is logged on" or "Run whether user is logged on or not" depending on your needs.
  + For higher privilege tasks, check "Run with highest privileges."

1. **Set Triggers**:

* **Triggers Tab**: Go to the "Triggers" tab and click on "New" to set a trigger.
* **Choose a Trigger**:
  + You can set the task to run on a schedule (daily, weekly, monthly), at startup, at logon, or in response to an event.
  + Configure the specific settings based on the trigger you choose (e.g., time of day, days of the week).
* **Enable or Disable**: You can temporarily disable the trigger by unchecking "Enabled."

1. **Configure the Action**:

* **Actions Tab**: Go to the "Actions" tab and click on "New" to specify what action the task should perform.
* **Choose an Action**:
  + **Start a Program**: This is the most common action. Specify the path to the executable or script you want to run.
  + **Send an Email**: This option is deprecated in recent versions of Windows.
  + **Display a Message**: This option is also deprecated in recent versions of Windows.
* **Program/Script**:
  + For "Start a Program," browse and select the program or script you want to run.
  + If the script requires arguments, you can add them in the "Add arguments" field.
* **Start In**: Specify the working directory if needed.

1. **Set Conditions**:

* **Conditions Tab**: Go to the "Conditions" tab to set additional conditions under which the task will run.
* **Power Conditions**:
  + For example, you can choose to run the task only if the computer is idle or on AC power.
* **Network Conditions**:
  + Specify if the task should only run when connected to a specific network.

1. **Additional Settings**:

* **Settings Tab**: Go to the "Settings" tab to configure additional settings for the task.
* **Allow Task to Be Run on Demand**: Check this if you want to be able to run the task manually.
* **Stop the Task if It Runs Longer Than**: Set a time limit for how long the task is allowed to run.
* **If the Task Fails**: Choose what happens if the task fails, such as retrying after a set interval.

1. **Review and Save**:

* **Review Settings**: Once all configurations are made, review your settings across all tabs.
* **Save the Task**: Click "OK" to save the task. If you chose "Run whether user is logged on or not," you might be prompted to enter your user account password.

1. **Test the Task**:

* **Run Task Manually**: After creating the task, you can test it by selecting it in the Task Scheduler Library, right-clicking, and choosing "Run."
* **Check History**: If there are issues, check the "History" tab for logs on the task’s execution.

**Commands To Run PowerShell:**

**1. Open PowerShell**

* Basic Access:
  + Press Windows + S, type "PowerShell", and press Enter.
* Administrative Access:
  + Press Windows + X, select "Windows Terminal (Admin)" or "Windows PowerShell (Admin)".

**2. Understand the PowerShell Environment**

* PowerShell Prompt: The command-line interface where you enter commands.
* Execution Policy: Controls the running of scripts. Check it with:

🡪**Get-ExecutionPolicy --** command in PowerShell shows the current **execution policy** for your PowerShell environment. The execution policy determines the level of restriction placed on running scripts and configuring the PowerShell environment.

**Execution Policy Levels**

Here are the common execution policy levels in PowerShell:

1. **Restricted**: No scripts can be run. This is the default setting on Windows client computers.
2. **AllSigned**: all scripts to be digitally signed by a trusted publisher before they can be run, regardless of their source (whether they are created locally or downloaded from the internet).
3. **RemoteSigned**: policy requires that all scripts and configuration files downloaded from the internet must be signed by a trusted publisher before they can be run. However, scripts and configuration files created locally on your machine do not need to be signed.
4. **Unrestricted**: No restrictions; all scripts can be run. However, there is a warning prompt for scripts downloaded from the internet.
5. **Bypass**: No restrictions; scripts run without any warnings or prompts.
6. **Undefined**: No policy is set in the current scope. This means that the policy is inherited from a higher scope.

* **Get-ComputerInfo** – This Command retrieves detailed information about the computer system, including hardware, operating system, and other system-related information.
* **Get-CimInstance -ClassName Win32\_OperatingSystem** – This Command get the details of the OS
* **Get-NetIPAddress** –Provide the details of IP address
* **Get-Help** – This Command is used to get the help for cmdlet
* **Get-Command** – This Command is used to get the list of all the command
* **Get-ChildItem** – This Command is used to get the list of all the Files in a system
* **Get-ChildItem -Path C:\Path\To\Directory**—To List File in a Specific Directory
* **Set-Location -Path C:\Path\To\Directory** – To Change Directory
* **Copy-Item -Path C:\Source\File.txt -Destination C:\Destination\** -- Copying Files from One directory to another
* **Move-Item -Path C:\Source\File.txt -Destination C:\Destination\** -- Moving Files
* **Remove-Item -Path C:\Path\To\File.txt**
* **Get-Process** – Get the List of all the process in a system
* **Get-Process -Name processname** – get the specific process
* **Get-Service** – get the List of All the Services
* **Get-Service -Name servicename** --Get the Specific service
* **Start-Service -Name servicename** --Start the specified service
* **Stop-Service -Name Servicename** --Stop a specified Service
* **Get-HotFix** – get the installed Updates
* **Get-LocalUser** –get the info of Local user
* **New-LocalUser -Name "username" -Password (ConvertTo-SecureString "password" -AsPlainText -Force)** – To Create a New Local user

Example: New-LocalUser -Name "JohnDoe" -Password (ConvertTo-SecureString "P@ssw0rd123" -AsPlainText -Force) -FullName "John Doe" -Description "A new user account"

* **$myVar = “Nishat Anjum Abid Ali Hosangadi”** – To create a Variable
* **Write-Output $myVar** – To list created Variable
* **Get-Module -ListAvalible** -- used to list all modules that are installed on the system and available for use. This cmdlet provides information about modules that are available to be imported into your PowerShell session.
* **Import-Module ModuleName –** Importing the Module from the module available List
* **New-Item -Path "C:\Path\To\File.txt" -ItemType File** – CreatesNew file

Example: New-Item -Path "C:\Temp\MyFile.txt" -ItemType File

* **"Text content" | Out-File -FilePath "C:\Path\To\File.txt"** -- cmdlet sends output to a file. If the file doesn’t exist, it will be created.

Example: "Hello, World!" | Out-File -FilePath "C:\Temp\MyFile.txt"

* **Set-Content -Path "C:\Path\To\File.txt" -Value "Text content"** -- cmdlet writes content to a file, creating the file if it doesn’t exist.

Example: Set-Content -Path "C:\Temp\MyFile.txt" -Value "Hello, World!"

* **Add-Content -Path "C:\Path\To\File.txt" -Value "Additional text"** -- cmdlet appends content to a file, creating the file if it doesn’t exist.

Example: Add-Content -Path "C:\Temp\MyFile.txt" -Value "This is additional content."

**Summary**

* **New-Item**: Creates a new file.
* **Out-File**: Creates a file and writes output to it.
* **Set-Content**: Creates a file or overwrites content in an existing file.
* **Add-Content**: Appends content to a file or creates it if it doesn’t exist

**Linux Basic Commands**:

1. **`sudo su**`: Switches to the root user with administrative privileges.

2. **pwd:** Prints the current working directory.

3. **cd:** Changes the current directory to the specified path.

4. **cd ..:** Moves up one directory level.

5. **mkdir:** Creates a new directory.

6. **touch:** Creates an empty file or updates the timestamp of an existing file.

7. **echo:** Displays a line of text or outputs text to a file.

8. **cat:** Concatenates and displays the contents of a file.

9. **nano:** Opens a file in the Nano text editor for editing.

10. **ps:** Displays a snapshot of current running processes.

11. **top:** Displays real-time system resource usage and running processes. After using Top command to get back to Terminal use ‘q’command

12. **vi:** Opens a file in the Vi text editor for editing.

**Save and Exit**:

* Press Esc to enter command mode.
* Type :wq and press Enter to save your changes and exit.

**Exit Without Saving**:

* Press Esc to enter command mode.
* Type :q! and press Enter to exit without saving any changes.

13. **ls:** Lists the contents of a directory.

14. **rm:** Removes files or directories.

15. **rm -r:** Recursively removes directories and their contents.

16. **head filename:** Displays the first few lines of a file.

17. **free:** Shows the system's memory usage.

18. **df -h:** Display sizes in human-readable format (e.g., KB, MB, GB).

19. **du -h:** Shows disk usage of files and directories in a human-readable format.

20. **free -h:** Displays memory usage in a human-readable format.

21. **wget:** Downloads files from the web via HTTP, HTTPS, or FTP.

22. **history:** Displays the list of previously executed commands.

23. **kill:** Sends a signal to terminate a running process.

24. **curl http://checkip.amazonaws.com:** Retrieves the public IP address of the system.

25. **ifconfig:** Displays network interface configurations (deprecated; use `ip a` instead).

26. **uname:** Displays system information, such as the operating system and kernel version.

27. **cp source\_file destination\_file:** Copies files or directories from one location to another.

28. **mv old name new\_name:** Moves or renames files or directories.

29. **rmdir:** Removes empty directories.

30. **yum install:** Installs software packages using the YUM package manager.

31. **yum update:** Updates all installed packages to their latest versions using YUM.

32. **df -h –total:** Display a grand total at the end of the output.

33**. grep 'pattern' filename:** searches for patterns within files.

34. **find /path -name filename:** Searches for files and directories based on various criteria.

35. **uniq filename:** Report or Omit Repeated Lines

36. **sort filename:** Sort Lines of Text Files

37. **wc filename:** Word, Line, Character, and Byte Count

38. **yum install package\_name**: is used to install a software package on a Linux system

39. **head -n 10 filename:** First Ten lines of fille content

40**. tail filename:** View the Last Few Lines of a File

41. **tail -n 10 filename:** To specify the number of lines

42. **whoami:** Display the Current Username

43. **hostname**: Display the Hostname of the System

44.**uptime:** Show How Long the System Has Been Running

45. **top**: Shows system resource usage and running processes in real-time.

46. **vim:** A powerful text editor package

47. **httpd:** The Apache HTTP Server, used to host web pages, Package.

**User and Group management in Linux:**

**User:** A user is an **individual or entity that interacts with a computer system, application, or service, typically through a unique account**

User can **create new User Account** by using **useradd** or **adduser** command followed by **username**

**User** can **perform task, own files, Utilize the features provided by system, Access resources, Log into system.**

**Types of Users:**

1. **Root User (Superuser)**:
   * The root user has **unrestricted access to all commands and files on the system.** It can perform **administrative tasks such as installing software, managing user accounts, and changing system configurations.**
2. **Regular Users**:
   * Regular users have limited permissions and can only access files and directories they own or have been granted permission to use. They can also execute commands but not administrative tasks.

**Group:**

Group is a Collection of Users; Groups are **used to manage and control the access permission for Multiple users Simultaneously.**

**Types of Groups:**

1. **Primary Group:**

When a user is created in Linux, they are automatically assigned to a **primary group**. This group is typically named after the user.

* **Ownership of Files**: Any files or directories that the user creates will be associated with their primary group.

**Example**:

* Suppose you create a user named nishat. By default, Linux will create a group also named nishat, and this group will be the primary group for the user.
* If nishat creates a file, say file.txt, the ownership will look like this

-rw-r--r-- 1 nishat nishat 0 Aug 28 10:00 file.txt

Here, the first nishat is the username (file owner), and the second nishat is the primary group.

**Secondary Groups**

* **What They Are**:
  + A user can also belong to **secondary groups** in addition to their primary group. These groups give the user additional access to resources, files, or directories that are shared among members of that group.
  + Secondary groups provide extra permissions without changing the user's primary group.
* **Example**:
  + Suppose you have a group called devs that multiple users belong to, including nishat.
  + This means nishat can access files and directories that are assigned to the devs group, even though her primary group is nishat.
  + If another user named alice belongs to the devs group and creates a file that is group-owned by devs, then nishat can also access it if permissions allow:
  + -rw-r--r-- 1 alice devs 0 Aug 28 10:00 shared\_file.txt
  + Here, alice is the file owner, but devs is the group owner. Since nishat is in the devs group, she has access to this file.

**Why It Matters**

* **Primary Group**: Determines the default group ownership for any files and directories a user creates.
* **Secondary Groups**: Allow users to share access to files and directories with other users who belong to the same secondary group.

**In Summary**

* **Primary Group**: Automatically assigned when a user is created; associated with everything the user creates.
* **Secondary Groups**: Additional groups a user can be a part of, granting them extra access to shared resources.

**User Management Files:**

* **/etc/passwd**: This file stores user account information, including usernames, user IDs (UIDs), and home directories.
* **/etc/shadow**: This file contains encrypted user password information.

**Group Management Files:**

* **/etc/group**: This file contains group information, including group names, group IDs (GIDs), and group members.

**User and Group IDs**

* **UID (User ID)**: Each user is assigned a unique UID, which the system uses to identify the user.
* **GID (Group ID)**: Each group is assigned a unique GID, which the system uses to identify the group.

**To create a new user in Linux:**

**Step 1: Open the Terminal**

* Make sure you have root or sudo privileges to create a new user.

**Step 2: Create the User**

* Use the useradd command followed by the desired username. For example, to create a user named john:

**🡪 useradd john**

**Step 3: Set a Password for the User**

* After creating the user, set a password using the passwd command

🡪**passwd john**

**You'll be prompted to enter and confirm the new password for the user.**

**Optional: Additional User Setup**

1. **Create a Home Directory (if not automatically created)**
   * Use the -m option to create a home directory for the user:

🡪**useradd -m john**

1. **List the /home Directory:**

* To check if the home directory was created, list the contents of the /home directory
* ls /home

If the directory is not present, the home directory was not automatically created, and you can create it manually using the -m option with useradd.

**Steps to Check the Primary Group:**

1. **Create the User:**
   * When you create a new user, Linux usually creates a primary group with the same name as the user by default.
2. **Check the Primary Group Using getent:**
   * To verify whether the primary group was created, use the getent command followed by the group name:
   * 🡪**getent group username**

**Expected output : john:x:1001:**

This output indicates that the group john exists, with a group ID (GID) of 1001. The x represents a placeholder for the group password (not used in modern systems).

1. **Check the Group Information in /etc/group:**

* Alternatively, you can manually search for the group in the /etc/group file by running
* 🡪 **grep john /etc/group**
* **This will display the line from /etc/group related to the user john.**

**Summary:**

* Use getent group username to check if the primary group for a user exists.
* The command should return a line with the group’s name and GID if the group was successfully created.

**Steps to Create a Secondary Group:**

1. **Create the Group:**
   * Use the groupadd command to create the group.
   * 🡪**groupadd Deploy**
2. **Add a User to the Secondary Group:**

* Once the group is created, you can add an existing user to this group using the **usermod** command with the **-aG** options.
* 🡪 **usermod -aG groupname username**

**Verification:**

1. **Check the Group Membership:**
   * To verify that the user has been added to the secondary group, you can use the groups command:

🡪**groups john**

**Expected Outcome**

**🡪** **john : john Deploy --** This indicates that john is in both the john (primary) group and the Deploy (secondary) group**.**

1. **Check /etc/group:**

* You can also verify by looking at the /etc/group file:
* **🡪 grep groupname /etc/group --** When you run the command grep groupname /etc/group, grep searches the /etc/group file for any lines that contain the word "groupname" and displays those lines.
* **This will show you all users who are members of the Deploy group.**

**Assign the User to a Specific Group:**

1. To assign the user to a specific group, use the -g option

* useradd -g groupname john

**Summary:**

* Create a group with sudo groupadd groupname.
* Add a user to the secondary group using sudo usermod -aG groupname username.
* Verify group membership with the groups username command.

**Switching Users (su Command):**

The su command (short for "substitute user" or "switch user") allows you to switch from your current user account to another user account. When you switch users, you effectively become that user, with all their permissions and environment settings.

* 1. **Switching to Another User**

**🡪** **su username**

You’ll be prompted to enter the password for user. After that, you’ll be logged in as user we want.

* 1. **Switch to the Root User (Superuser):**

**🡪su -** These switches you to the root user, provided you know the root password. The - option loads the root user's environment variables as well.

* 1. **Exit the Switched User Session:**
* To return to your original user after switching, simply type

🡪**exit**

**Using sudo for Elevated Privileges:**

The sudo command (short for "superuser do") allows a permitted user to execute a command as the root user or another user, as specified in the /etc/sudoers file. sudo is commonly used to execute a single command with superuser privileges without needing to switch users entirely.

**1.Execute a Command as Root:**

**🡪sudo command i.e 🡪** **sudo apt-get update**

This command will run the apt-get update with root privileges. You’ll be prompted for your own password, not the root password.

**2.Run a Command as Another User:**

**🡪** **sudo -u username command i.e🡪** **sudo -u username command**

This command will list the contents of john's home directory as if you were john.

**Key Differences:**

* **su Command:**
  + Used to switch to another user account.
  + Requires the password of the account you're switching to.
  + Once switched, all subsequent commands are run as the switched user until you exit.
* **sudo Command:**
  + Used to run a single command with elevated privileges (usually root).
  + Requires your own password (assuming you have sudo privileges).
  + Only the specified command is run with elevated privileges; subsequent commands are run as your regular user.

**Summary:**

* **su**: Switches your session to another user account, effectively becoming that user until you exit.
* **sudo**: Runs a single command with elevated privileges without fully switching users, making it safer and more convenient for administrative tasks.

**FILE PERMISSION AND CONTROL:**

In Linux, file permissions and ownership control who can read, write, or execute files and directories. Here’s a breakdown of how to use chmod, chown, and chgrp commands to manage permissions and ownership:

**1. File Permissions: chmod**

The chmod (change mode) command changes the permissions of a file or directory. Permissions are represented by three types of access:

* Read (r): Allows viewing the contents of a file or listing the contents of a directory.
* Write (w): Allows modifying the contents of a file or creating/deleting files in a directory.
* Execute (x): Allows executing a file as a program or script, or accessing a directory.

Permissions are set for three categories of users:

* Owner (u): The user who owns the file.
* Group (g): Users who are members of the file’s group.
* Others (o): All other users.

Examples:

**Set Permissions Using Symbolic Mode:**

**🡪chmod u+r file.txt** # Add read permission for the owner

**🡪chmod g-w file.txt** # Remove write permission for the group

**🡪chmod o+x file.txt** # Add execute permission for others

**🡪chmod u=rwx,g=rx,o=r file.txt** # Set permissions: owner can read/write/execute, group can read/execute, others can read

**Set Permissions Using Numeric Mode:**

Permissions are represented by a three-digit octal number:

* 4 = read (r)
* 2 = write (w)
* 1 = execute (x)

Each digit represents permissions for owner, group, and others respectively

🡪**chmod 755 file.txt** # Set permissions: owner can read/write/execute, group and others can read/execute

🡪**chmod 644 file.txt** # Set permissions: owner can read/write, group and others can read only

**File Ownership: chown**

The chown (change owner) command changes the ownership of a file or directory. You can change the owner, the group, or both.

**Examples:**

**Change File Owner:**

**🡪** **sudo chown newowner file.txt**

**Change File Group:**

**🡪 sudo chown :newgroup file.txt**

**Change Both Owner and Group:**

**🡪** **sudo chown newowner:newgroup file.txt**

**Group Ownership: chgrp**

**The chgrp (change group) command changes the group ownership of a file or directory. It is a simpler way to change only the group without affecting the owner.**

**Change Group Ownership:**

**🡪** **sudo chgrp newgroup file.txt**

**Summary:**

* chmod: Changes file permissions.
  + Symbolic Mode: chmod u+r, chmod g-w
  + Numeric Mode: chmod 755, chmod 644
* chown: Changes file ownership.
  + Change Owner: sudo chown newowner file.txt
  + Change Group: sudo chown :newgroup file.txt
  + Change Both: sudo chown newowner:newgroup file.txt
* chgrp: Changes the group ownership.
  + Change Group: sudo chgrp newgroup file.txt

These commands are crucial for managing file security and access control in a Linux system.