**Day-12 Task for #90DaysofDevops  
#owner-vishal saxena**

**Cheat-sheet for the commands of Basic Linux, Adv Linux, Bash Scripting, Git & Github basic and advance**

1. File and directory management:

* + **ls**: list files and directories in the current directory
  + **cd**: change directory
  + **pwd**: print the current working directory
  + **mkdir**: create a new directory
  + **touch**: create a new file
  + **cp**: copy files and directories
  + **mv**: move or rename files and directories
  + **rm**: delete files and directories
  + **chmod**: change file permissions
  + **chown**: change file ownership
  + **echo:** Prints a message to the console
  + **less:** Allows you to view the contents of a file one page at a time
  + **grep:** Searches for a pattern in a file or a stream of text
  + **sed:** Modifies the contents of a file using a script
  + **awk:** Processes text files and data streams
  + **find:** Searches for files and directories
  + **tar:** Creates and extracts archive files

1. System Information and monitoring:  
   * **uname -a** : display the kernel version, hostname
   * **free -m** : display the amount of free and used memory
   * **df -h** : display the amount of disk space used and available
   * **top** : display the processes running on the system
   * **ps aux** : display the processes running and their status
   * **uptime** : show the system uptime and load averages
   * **head :** shows the first few lines of the file or output ( default it shows top 10 lines )
   * **tail :** shows the last few lines of the file or output ( default it shows last 10 lines )
   * **cron :** It allows users to schedule tasks (also called "cron jobs") to run automatically at specified times and dates.
   * **crontab :** schedules commands to be executed automatically
   * **kill :** sends a signal to a process to terminate it
2. Networking:
   * **ifconfig**: display network interface configuration
   * **ping**: test network connectivity
3. Users and Groups:
   * **whoami**: display the current user
   * **id**: display user and group information
   * **adduser**: add a new user to the system
   * **userdel**: delete a user from the system
   * **addgroup**: add a new group to the system
   * **groupdel**: delete a group from the system
4. Package management:
   * **apt-get**: package manager for Debian-based systems
   * **yum**: package manager for Red Hat-based systems

**Git commands :**

1. Basic Git commands:
   * **git init:** initialize a new Git repository
   * **git clone**: clone an existing repository
   * **git add**: add changes to the staging area
   * **git commit**: commit changes to the repository
   * **git status**: check the status of the repository
   * **git log**: view the commit history
   * **git checkout [branch name]:** Switches to the branch with the given name
   * **git checkout -b [branch name]:** Creates a new branch with the given name and switch to it
2. Git branching:
   * **git branch**: view and manage branches
   * **git branch [branch name]:** Creates a new branch with the given name
   * **git branch -d [branch name]:** Deletes the branch with the given name
   * **git checkout**: switch to a different branch
   * **git merge**: merge changes from one branch into another
3. Git on GitHub:
   * **git push**: push changes to a remote repository on GitHub
   * **git pull**: pull changes from a remote repository on GitHub
   * **git fork**: create a fork of a repository on GitHub

1. Advanced Git commands:
   * **git diff**: view differences between commits, branches, etc.
   * **git reset**: undo changes to the repository
   * **git rebase**: reorganize commits in a branch

5. Git Synchronizing

* + **git fetch [remote]**: fetch changes from a remote repository
  + **git fetch --all**: fetch changes from all remote repositories
  + **git remote -v**: list all configured remote repositories
  + **git merge [remote]/[branch]**: merge changes from a remote branch into the current branch
  + **git pull [remote] [branch]**: fetch and merge changes from a remote repository in one command
  + **git push [remote] [branch]**: push changes to a remote repository
  + **git push --set-upstream [remote] [branch]**: set the current branch to track a remote branch
  + **git remote add upstream [remote]**: add the original repository as an "upstream" remote
  + **git fetch upstream**: fetch changes from the upstream repository
  + **git merge upstream/[branch]**: merge changes from the upstream repository into the current branch
  + **git push**: push the updated local repository to the fork
  + **git status**: check the status of the repository and identify conflicting files
  + **git diff**: view the differences between the conflicting changes
  + **git merge --abort**: abort the merge and revert to the previous state
  + **git add [file]**: stage the resolved changes
  + **git commit**: commit the resolved changes

Note: It is a good practice to always fetch and merge changes from the remote repository before pushing changes, to avoid conflicts and ensure that your local repository is up-to-date. Also, make sure to check the status of your repository frequently when synchronizing with remote repositories.

-----------------------**THE END**-----------------------