**Devops:**

**To start learning DevOps as a beginner, you need a structured approach covering foundational topics and practical tools. Below is a step-by-step guide:**

**1. Linux: Bash:**

* + File system: ls, cp, mv, rm
  + permissions: chmod, chown
  + processes: ps, top, kill
  + packages: apt, yum

**2. Networking concepts: Wireshark:**

* + OSI and TCP/IP models
  + IP addressing and subnetting
  + DNS and DHCP
  + HTTP, HTTPS,FTP, and SSH
  + Firewalls and security groups
  + ping, traceroute, and netstat

**3. GIT:**

* + Clone, commit, push, and pull
  + Branching and merging
  + Resolving merge conflicts
  + working with remote repositories

**4. Programming language: Python**

* Python syntax and data structure
* Modules and packages
* write and execute Python scripts
* work with files
* handle errors
* write automation scripts

**5. Cloud Providers: AWS:**

* Launch, configure, and mange virtual servers
* store and manage data
* manage users, groups, and roles
* setup and manage isolated networks

**6. Containerization: Docker:**

* Create Docker images
* starting, stopping and managing containers
* write Docker files
* Run multi-container apps using Docker compose

**7.CI/CD: Jenkins:**

* create and manage Jenkins pipelines
* write Jekinsfiles
* Integrate automate tests
* Automate the build process
* Automate the deployment

**8. Orchestration and management: Kubernetes:**

* overall architecture
* key components
* Manage resources
* scale application
* Networking model in Kubernetes

**9. Networking and Infrastructure as services: Nginx:**

* Configure Nginx as a reverse proxy
* configure it to act as a forward proxy
* caching strategies
* configure firewalls and security groups

**10. Configuration management: Ansible:**

* write Ansible playbooks
* Use roles and modules
* Manage variables and templates

**11. Infrastructure as code (IAC): Terraform:**

* Basic Concepts
* Terraform Configuration files
* Terraform modules
* Advanced concepts

**12. Monitoring and Logging: Prometheus and Grafana**

* Architecture and data model
* collect metrics
* write queries
* set up alerts

**Linux**

**Introduction:**

The beginning to 1969 where Ken Thompson and Dennis Ritchie of Bell Laboratories developed the UNIX operating system.  It was later rewritten in C to make it more portable and eventually became a widely used operating system. Then in 1991, a young fellow named Linus Torvalds started developing the Linux kernel.

**Linux Distribution**

1. **Debian**: Debian is an operating system composed entirely of free and open-source software. It’s widely known and has been in development for over 20 years. If you want a good "core" operating system, this is the one for you.
2. **Arch Linux**: Arch is a lightweight and flexible Linux distribution driven 100% by the community. Similar to Debian, Arch uses a rolling release model so incremental updates eventually become the Stable release. There’s a bit of a learning curve, but for the hardcore Linux users, this is a great choice.
3. **Red Hat Enterprise Linux**: Red Hat Enterprise Linux commonly referred to as RHEL is developed by Red Hat. RHEL has strict rules to restrict free re-distribution although it still provides source code for free. RHEL-based operating systems will differ slightly from the Debian-based operating systems, most noticeably in package management. If you decide to go with RHEL it’s probably best if you know you’ll be working with it.
4. **Fedora**: Backed by Red Hat, the Fedora Project is community driven containing open-source and free software. Red Hat Enterprise Linux branches off Fedora, so think of Fedora as an upstream RHEL operating system. Eventually RHEL will get updates from Fedora after thorough testing and quality assurance. Fedora is great if you want a Red Hat based operating system without the price tag. Recommended for desktop and laptop.
5. **Ubuntu**: One of the most popular Linux distributions for personal machines is Ubuntu. Ubuntu also releases its own desktop environment manager Unity by default. Great for any platform, desktop, laptop and server.
6. **Linux Mint**: Linux Mint is based off of Ubuntu. It uses Ubuntu’s software repositories so the same packages are available on both distributions. If you prefer a lighter distro than Ubuntu, you may be interested in Linux Mint. Great user interface, great for beginners and less bloated than Ubuntu.
7. **Gentoo**: Gentoo offers ridiculous flexibility with the operating system at a price. It’s made for advanced users who don’t mind getting their hands dirty with the system. If you’re just getting started with Linux and want to take a more difficult path
8. **OpenSUSE**: openSUSE Linux is created by the openSUSE Project. A community that promotes the use of Linux everywhere, working together in an open, transparent and friendly manner as part of the worldwide Free and Open-source Software community. openSUSE includes everything you need to enjoy the Internet worry free of viruses/spy-ware and to live out your creativity, be it with your photos, videos, music or code.

**Command Line**

**Fundamentals command line, navigating files, directories**

**Shell**: The shell is basically a program that takes your commands from the keyboard and sends them to the operating system to perform. The shell program uses bash (Bourne Again shell), almost all Linux distributions will default to the bash shell. There are other shells available such as ksh, zsh, tsch, etc. $date; $whoami

1. **Pwd (print working directory):** Everything in Linux is a file. Every file is organized in a

hierarchical directory tree. The first directory in the filesystem is aptly named the root directory. The root directory has many folders and files which you can store more folders and files, etc. Ex:

/  
|-- bin  
| |-- file1  
| |-- file2  
|-- etc  
| |-- file3  
| `-- directory1  
| |-- file4  
| `-- file5  
|-- home  
|-- var

The pwd to see where you are, you can use the pwd command, this command means “print working directory” and it just shows you which directory you are in. Ex: $pwd

1. **Cd (Change Directory):** There are two different ways to specify a path, with absolute and

relative paths.

* **Absolute path**: This is the path from the root directory. The root directory is commonly shown as a slash.  Every time your path starts with / it means you are starting from the root directory
* **Relative path**: This is the path from where you are currently in filesystem

**Shortcuts for cd to help out:**

* . (current directory). This is the directory you are currently in. ex: cd .
* .. (parent directory). Takes you to the directory above your current. Ex: cd ..
* ~ (home directory). This directory defaults to your “home directory”. Such as /home. ex: cd ~ (or)
* Only if we type cd it will redirect to home directory. ex: cd
* - (previous directory). This will take you to the previous directory you were just at. Ex: cd -

1. **Ls(List Directories**): The ls command will list directories and files in the current directory.

**ls -a** : Filenames that start with . are hidden, you can view them

**ls -l** :  -l for long, this shows a detailed list of files in a long format. This will show you detailed information, starting from the left: file permissions, number of links, owner name, owner group, file size, timestamp of last modification, and file/directory name.

total 40

drwxr-xr-x 3 raji raji 4096 Oct 25 15:49 Desktop

drwxr-xr-x 2 raji raji 4096 Oct 16 19:20 Documents

* ls -la or ls -al
* ls -R: recursively list directory contents
* ls -r: reverse order while sorting
* ls -t: sort by modification time, newest first

1. **Touch:**Touch allows you to the create new empty files.

touch file\_name

Touch is also used to change timestamps on existing files and directories. Give it a try, do an ls -l on a file and note the timestamp, then touch that file and it will update the timestamp.

1. **File**: To find out what kind of file a file is, you can use the file command. It will show you a description of the file’s contents. Ex: file file\_name
2. **Cat**: A simple command to use is the cat command, short for concatenate, it not only displays file contents but it can combine multiple files and show you the output of them. Ex: cat file\_name ; cat file\_name file\_name
3. **Less**: f you are viewing text files larger than a simple output, less is more. The text is displayed in a paged manner, so you can navigate through a text file page by page.Ex: less file\_name.

**Use the following command to navigate through less:**

* q - Used to quit out of less and go back to your shell.
* Page up, Page down, Up and Down - Navigate using the arrow keys and page keys.
* g - Moves to beginning of the text file.
* G - Moves to the end of the text file.
* /search - You can search for specific text inside the text document. Prefacing the words you want to search with /
* h - If you need a little help about how to use less while you’re in less, use help.

1. **History**: In your shell, there is a history of the commands that you previously entered, you can actually look through these commands. This is quite useful when you want to find and run a command you used previously without actually typing it again. Ex: $ history. (or) Use!!. If you typed cat file1 and want to run it again, you can actually just go !! and it will run the last command you ran. (or) Use!!. If you typed cat file1 and want to run it again, you can actually just go !! and it will run the last command you ran. Clear Our terminal is getting a little cluttered do a little cleanup, use the clear command to clear up your display. Ex: $clear
2. **CP(Copy):** Much like copy and pasting files in other operating systems. Ex: cp filename directory name. You can copy multiple files and directories as well as use wildcards. A wildcard is a character that can be substituted for a pattern-based selection, giving you more flexibility with searches.

* the wildcard of wildcards, it's used to represent all single characters or any string. Ex: cp \*.txt Desktop-> it will copy all .txt file to desktop folder.
* ? used to represent one character
* [] used to represent any character within the brackets
* $ cp -r folder name directory path-> it will copy all folder to given directory path.
* $cp -I filename directory path-> where -I is the interactive option. It prompts the user for confirmation **before overwriting a file** in the destination. Type y to overwrite the file.
* Type n to skip the overwrite.

1. Mv(Move): Used for moving files and also renaming them. Quite similar to the cp command in terms of flags and functionality.

* You can rename files like this: $ mv oldfile newfile
* Or you can actually move a file to a different directory:$ mv file2 Directory name
* And move more than one file: $ mv file\_1 file\_2 /some directory
* You can rename directories as well: $ mv directory1 directory2
* Like cp, if you mv a file or directory it will overwrite anything in the same directory. So you can use the -i flag to prompt you before overwriting anything: mv -i directory1 directory2
* mv a file to overwrite the previous one. You can also make a backup of that file and it will just rename the old version with a ~: $ mv -b directory1 directory2

1. **mkdir(Make Directory**):  The mkdir command (Make Directory) is useful for that, it will create a directory if it doesn’t already exist. You can even make multiple directories at the same time. ex: $ mkdir books paintings

You can also create subdirectories at the same time with the -p (parent flag).

$ mkdir -p books/hemmingway/favorites

1. **rm(Remove):** The rm (remove) command is used to delete files and directories. Ex: $rm file\_name

* Ex: $rm -f file\_name where -f force option tells rm to remove all files, whether they are write protected or not, without prompting the user (as long as you have the appropriate permissions).
* $rm -I file\_name: where -i flag like many of the other commands, will give you a prompt on whether you want to actually remove the files or directories.
* $rm -r directory\_name: You can’t just rm a directory by default, you’ll need to add the -r flag (recursive) to remove all the files and any subdirectories it may have.
* $rmdir directory\_name: remove a directory with the rmdir command.

1. **Find**:  To find a specific one. there’s a command we can use for that, find!

* $find Desktop -name file.txt: where -name to used as option to specify the name of the file to search for. Desktop is the starting directory where the search begins.
* $find Desktop -type d -name file: where type is used to specify the directory d

1. **Help**: help, is a built-in bash command that provides help for other bash commands (echo, logout, pwd, etc). ex: help echo
2. **Man**: Man pages are manuals that are by default built into most Linux operating systems. They provide documentation about commands and other aspects of the system. Ex: man ls
3. **Whatis**: The whatis command provides a brief description of command line programs. The description gets sourced from the manual page of each command. Ex: whatis cat
4. **Alias**: Sometimes typing commands can get really repetitive, or if you need to type a long command many times, it’s best to have an alias you can use for that.

* Ex: alias ra=’ls -la’ Now instead of typing ls -la, you can type ra. It will won't save your alias after reboot.
* so you'll need to add a permanent alias in: ~/.bashrc
* you can remove aliases with the unalias command: $unalias ra

1. **exit**:  To exit from the shell, you can use the exit command ex: $exit or $logout

**Text: Basic text manipulation and navigation:**

1. **stdout(standard out):$**echo hello world > file.txt: The hello world is store in file.txt, where the > is used to operator overwrite the content of file.txt with hello word, if the file file.txt does not exist means it will create. if the file already exists, it s existing content will be replace.

* $echo hello world >> file.txt: >> it will append hello world to the end of file.txt.
* $ls -l /var/log >file.txt:->var/log is a standard directory in linux syatems that stores log files for the system application
* $>file.txt: nothing stores in the file.txt
* > is as output stream
* Echo is used to display output

1. Stdin(standard in) : we have stdin from devices like the keyboard, but we can use files, output from other processes and the terminal as well.

* $cat < file.txt > text.txt: where it will replace all content which is in file.txt to text.txt.
* < : is used as input steam.
* Cat: used for input

1. Stderr(standard error): The contents of a directory that doesn't exist on your system and redirect the output to the peanuts.txt file again.

* $ ls /fake/dir > file.txt: error message ls: cannot access /fake/directory: No such file or directory.
* We will have to use file descriptors. A file descriptor is a non-negative number that is used to access a file or stream. The file descriptor for stdin, stdout and stderr is 0, 1, and 2 respectively.
* $ ls /fake/directory 2> file.txt: You should see just the stderr messages in peanuts.txt.
* $ls /fake/dir >file.txt 2>&1: This sends the results of ls /fake/directory to the peanuts.txt file and then it redirects stderr to the stdout via 2>&1. The order of operations here matters, 2>&1 sends stderr to whatever stdout is pointing to. In this case stdout is pointing to a file, so 2>&1 also sends stderr to a file.  So if you open up that peanuts.txt file you should see both stderr and stdout.
* $ls /fake/dir &> file.txt:  shorter way to redirect both stdout and stderr to a file
* $ls /fake/dir 2> /dev/null: well you can also redirect output to a special file call /dev/null and it will discard any input.

1. Pipe and tee: plumbing

* $ls -la /etc: You should see a very long list of items, it's a little hard to read actually.
* $ls -la /etc |less: The pipe operator |, represented by a vertical bar, allows us to get the stdout of a command and make that the stdin to another process. In this case, we took the stdout of ls -la /etc and then *piped* it to the less command. The pipe command is extremely useful and we will continue to use it for all eternity.
* $ls | tee file.txt: You should see the output of ls on your screen and if you open up the peanuts.txt file you should see the same information!

1. Env(Environment):

* $echo $HOME: You should see the path to your home directory, mine looks like /home/raji.
* $echo $USER: should see the username
* $env: It's coming from your environment variables