AWS DevOps

* **AWS:** **A**mazon **W**eb **S**ervices is a comprehensive and widely used cloud computing platform provided by Amazon.com. It offers a vast array of services including computing power, storage options, networking, databases, machine learning, analytics, security and more. AWS enables businesses and individuals to access and utilize computing resources without the need to invest and maintain the physical infrastructure.
* **SDLC:** Software Development Life Cycle is a structured process used by software developers to design, develop, test, and deploy software applications. The main objective of SDLC is to produce high-quality software.
  + Requirements Gathering: Client need’s and requirements are collected and documented.
  + Analysis: Analyses the gathered requirements and need’s.
  + Design: The software architecture and design are created based on requirements and analysis. It involves the system architecture,
  + Data structures, interfaces and other technical specifications.
  + Implementation (Coding): In this phase, Developers write code according to the design specifications.
  + Testing: Once the code is developed, it undergoes various testing phases to identify and fix defects and bugs.
  + Deployment: After successful testing, The software is deployed to production environment. This Involves installing the software on user’s system or servers and making it available for users **(Live).**
  + Maintenance and Support: Even after Deployment, the software requires outgoing maintenance and support to address issues, implement updates, and to add new features.
* Additionally, different methodologies such as Agile, Waterfall or DevOps may be employed to manage the SDLC Process.
* **DevOps:** Addressing all the traditionl issues and implimenting automation at evry stage of software development life cycle by using automation tools

It is a set of practices that combines software Development and IT Operations. It amis to shorten the systems development life cycle and provides continuous delivery with high software quality. Simply DevOps aims to improve the quality, speed and reliability of software delivery. It also involves the use of various tools and technologies to support the DevOps culture and processes.

* + **Some of the commom DevOps Tools are**:
* **JENKINS**
* **DOCKER**
* **SONAR CLOUD**
* **ANSIBLE**
* **KUBERNATES**
* **TERRAFORM**
* **GIT etc….**
* **AWS DevOps:** The DevOps within the AWS cloud environment (Platform).
* **AWS Management Console** is a web based graphical user interface that allows you to manage and monitor your AWS resources. It provides a single interface to access and manage over 150 AWS services, including Amazon S#, Amazon EC2 and Amazon RDS. AWS console is accessible from any web browser and user friendly i.e, you can simply search for any services that you want to use. And for any help or information regarding AWS you can ask the AWS Assistant Bot (Amazon Q) which was an always-on generative AI assistant.
* **IAM:** **I**dentity and **A**ccess **M**anagement is a web service that helps you securely control access to AWS resources. It is created by the root user. With IAM the root user can centrally manage permissions that control, which AWS resources user can access. Simly IAM is used to control who is authenticated(signed in) and authorized(has permissions) to use resources.
  + **IAM role**: An IAM role is an IAM identity which *manages who has access to your AWS resources*. An IAM role is similar to an IAM user.
  + **IAM Policy**: is a document with a set of rules that defines permissions for an identity or resource in AWS.

**NOTE**: An IAM role with no IAM policy attached to it won’t have to access any AWS resources and an IAM policy that is not attached to an IAM role is of no use.

* ***How to create an IAM user with the IAM roles and policies?***

Sign-in to AWS console and search IAM in the search bar on the console home. After that, go to IAM and select user and assign a user name and then you have to give IAM role and policies (set permissions) as per your requirement and review the created user and press create once you are done with that. You can delete or change the given permissions any time with the root credentials.

* **EC2:** Amazon **E**lastic **C**ompute **C**loud is a web service that allows users to rent virtual servers on which to run their applications without investing in physical hardware. Simply those virtual servers are called **“INSTANCES”.**
* Key Features of EC2:
* **Instances:** is a virtual server. It is a core part in the AWS’s cloud Computing platform and provides user with scalable computing capacity. It can be customized based on the various parameters such as memory, computing power, networking capacity.
* **Amazon Machine Images**
* **EBS (Elastic Block Storage)**
* **Network & Security**
* **Load Balancing**
* **Auto Scaling**
* **Server**: which provides functionality to the device or a project.

Working process: CURD

* Create
* Read
* Update
* Delete
* **EC2 Instance creation**:
* **Types of Instances**:
* **Pricing**: AWS offers several pricing models to accommodate different usage patterns and business needs.
* **ON-Demand Pricing**: pay as you go. Suitable for applications with variable workloads and short-term projects.
* **Spot Request**: Allow user to bid the unused ec2 instance capacity at a potentially lower cost (like an offer)
* **Reserved Instances**: It offers cost saving up to 75% as compared to on-demand instances
* **Full Afferent**: One time payment
* **Partial Afferent**: half half payment
* **No Afferent**: Use first and Pay later

Linux

* **Linux** is an open-source operating system kernel that serves as the foundation for various Unix-like operating systems, commonly referred to as Linux distributions (or distros). Linux is highly customizable and is widely used in server environments, embedded systems, and personal computers.
* **Why DevOps professionals often prefer Linux?**

Compatibility with Tools (Many DevOps tools and platforms are developed primarily for Linux environments.)

Scripting and Automation (Linux provides powerful command-line interfaces and scripting capabilities through shells like Bash. DevOps tasks,

such as provisioning, configuration management, deployment automation, and system monitoring, can be efficiently

performed using shell scripts and command-line tools available in Linux)

Containerization and Orchestration (Linux offers scalability and performance advantages, allowing DevOps teams to efficiently manage large-scale

infrastructure and handle high workloads. Linux-based servers can be optimized for performance, resource

utilization, and reliability, meeting the demands of modern applications and services.)

* **Linux Commands** are instructions or directives given to the Linux operating system through the command-line interface (CLI) or terminal. These commands are used to perform various tasks such as managing files and directories, manipulating processes, configuring system settings, networking etc…

**NOTE: Linux commands are case sensitive**

* + Some of the Linux Commands:
* **ls** : List directory contents.
* **ll** : List all the files including the extension files and hidden files.
* **ls -l** : List directory with details.
* cd : Change directory.

**cd </file/path/>** : To open the directory (folder)

* **pwd** : Print working directory.
* mkdir : Make directory

**mkdir <filename>**: To create a new folder or directory .

* rm : Remove

**rm <file name>** : Remove files or directories.

**rm –rf <file name>** : To force remove.

* cp , mv : Copy , move

**cp </source/path/> </dest/path/>** : Copy files or directories.

**mv </source/path/> </dest/path/>** : Move or rename files or directories.

* **touch <file name>** : To create empty text file (.txt).
* **cat <file name>**  : To view the data in the text files.
* **df -h**  : To know the hard disk storage.
* **du -sh** : To know the folder storage.
* **top**  : To know the CPU utilization.
* **cd ..** : To come one step back.
* **clear** : To clear the screen. (work history obsent)
* **ctrl+l** : Is a shortcut to clear the screen. (work history present)
* sudo : Execute commands with superuser privileges.
* **sudo -s**  : To become a superuser.
* **exit** : To come back or to log out.
* apt : Package manager It is used to install, remove, and manage software packages.

**apt install** :To install the packages or softwares

**apt update** : To update the server. (refresh)

**apt upgrade** : To upgrade the packages in the server.

* **kill** : To terminate process.
* **ps** : Process status.
* **whoami** : To display the current username.
* chmod , chown : Change mode, change owner

**chmod <mode> <file­\_name>** : To change the mode of the file.

**chown <option> <owner\_name> <file\_name>** : To change the owner of the file.

* systemctl : System control.

**systemctl start <service\_name>** : To start a service

**systemctl stop <service\_name>** : To stop a service

**systemctl restart <service\_name>** : To restart a service

**systemctl enable <service\_name>** : To enable a service

**systemctl disable <service\_name>** : To disable a service

**systemctl status <service\_name>** : To check the status

* **lsblk** : List information about all available block devices.
* find : To find the file.

**find / -name <file­\_name>** : To find the file.

* wget : To download from net

**wget <url>** : To download the data or a packages from the net by pasting the url.

* tar : To create and manipulate tar archives.

**tar –zxvf <tar\_file>** : To untar the tar file.

* zip/unzip : To create and extract the files.

**zip –r <filename.zip> <file names to zip>** : Command to zip the files.

**unzip <.zip\_file­\_name>** : Command used to extract the zip file.

**>filename** is to clear the complete date inside the extension file.

GIT & GitHub:

* **Git** is a distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
* It's an open-source tool that tracks changes in source code during software development.
* With Git, developers can work collaboratively on projects, track changes, revert to previous versions, and merge changes made by different team members.
* **GitHub** is a web-based platform for version control using Git.
* It offers hosting for software development projects, enabling collaboration between developers, code management, and various project management features

**First time:**

* git init
* git add .
* git commit -m "comment"
* git branch -M (branch name)
* git remote add origin (github repo link)
* git push -u origin (branch name)

**Changes:**

* git init
* git status
* git add .
* git commit –m “comment”
* git push

**git log** is a command used to display the commit history of a repository.

**SYNTAX**:

* git log
* git log --oneline

It displays

* Commit hash
* Author's name and email
* Date and time of the commit
* Commit message
* **This is the difference between the git log and git log –oneline**



* **git pull** is a Git command used to fetch the latest changes from a remote repository and integrate them into the current branch of your local repository. Essentially, it performs two actions: **git fetch** followed by **git merge**
* **git clone** command is used to create a copy of an existing Git repository. It fetches the entire repository, including all branches, commits, and files, and sets up a new local repository on your machine. This is particularly useful when you want to start working on a project that already exists in a remote Git repository.

**SYNTAX:**

For local repo : git clone <repository\_url>

For instance : git clone <repository url> <file name that u want to save this clone the repo>

* **git fetch** command is used to retrieve new commits from a remote repository without merging them into your local branches. It's essentially a way to update your local repository with the latest changes from the remote repository without automatically merging them into your working branch.
* **git merge is** a command used to merge changes from one branch into another. It combines the changes made in one branch with another branch, resulting in a new commit that reflects the combined history of both branches**.**

Web-Servers

* **web server** is a software application or hardware device responsible for serving web content to clients over the internet.
* It plays a critical role in enabling access to web resources and facilitating communication between clients and servers on the World Wide Web.

Some of the popular web-server software applications:

* Nginx
* Apache
* Node.js
* Caddy
* LiteSpeed
* Microsoft IIS (Internet Information Services) etc..
* **Nginx** is an open source web server known for its high performance, scalability, and efficiency.

syntax to install nginx: **apt install nginx –y**

port number for nginx: **80**   
Path of nginx: **/var/www/html/**

You will find the **web content** (**.html**) file inside this path.

* **Apache** is an open-source web server software developed and maintained by the Apache Software Foundation. It is one of the most widely used web server applications globally.

Syntax to install apache: **apt install apache2 –y**

Port number for apache2: **80**

Path of apache2: **/var/www/html/**

You will find the **web content** (**.html**) file inside this path.

* Deploying a webpage using these web-server softwares in EC2 instance:

Steps-

1. Launch one EC2 instance.
2. Connect to the server and become a root user **sudo –s**  and update the server **apt update .**
3. Now, install the web-server software applications either Apache (**apt install apache2 –y**)

or nginx (**apt install nginx –y**)

1. Go to the web content path **cd /var/www/html**
2. We will find index.html file there.
3. Either we can edit that index.html file with our webpage code given by developers using **vim** or **nano** or we can clone the repository from GitHub (**git clone <repository\_url>**) or we can download the web files using url (**wget <webfile\_url>**).
4. If we download the webpage from the web using wget it will be downloaded as a .zip file. So we have to unzip that file to deploy. For that we need to install the unzip (**apt install unzip –y**). after the installation is done we have to unzip that .zip file by **unzip <.zip\_file­­\_name> .**
5. After the .zip file was extracted we have to remove the .zip file (**rm <.zip\_file\_name>**) just for storage purpose.
6. Now, the extracted files are stored in a new folder in html folder. Open that new folder (**cd <folder­\_name>**) So we have to move all the files in this new folder to the html folder i.e, one step back **mv \* ..**
7. Now, we can remove that empty new folder since we have moved all the files in that folder there is no use of that folder anymore. **rm –rf <new­\_folder\_name>**
8. We have to do the same process except the unzip when we clone from the GitHub.
9. Now, to access this webpage in the internet we have to add the port number 80 (both the nginx and the apache port number is 80).
10. To add the port number: **select the instance** and go to **security** and then open **security groups**

Choose the **edit inbound** **rules** and **add port 80** in it.

1. Now, copy the public id and past it in the browser with :80 at the end (**public\_ip:80**).
2. Hence the web-page was deployed.

Elastic IP

* **Elastic IP server capacity**: In any web-application or a streaming application, if the users or the work load increases then we have to increase the EC2 instance capacity to reduce the traffic (to decrease the work load). So that more users can access the application.
* Before doing that we have to inform the team members about this. And take the approval from the networking team, application team and manager.
* **Procedure**:

1. Select the EC2 instance in which the application was deployed.
2. **Stop** the server in instance state options.
3. After the instance was stopped, go to **actions** and select the **instance settings** now select the **instance type**.
4. Now, change the instance type as per requirement and **apply changes**.
5. Start the instance now and check weather the application in the server is running stage or not.
6. Check the following commands: **df –h**

**top**

**systemctl status <application­\_name>**

**systemctl restart <application\_name>**

* Every time after stopping and stating the instance the public ip of the instance will be changed. It will be a problem for all the users who are having old ip address.
* **Elastic IP server** is the solution for that problem.
* **Elastic IP** is a service in the EC2 which provides the fixed ip address to the instance.

it is a paid service. It will charge even the instance is not in running stage.

Therefore, it's advisable to use them wisely and release them when not in use, to avoid unnecessary costs.

Each AWS account contains only 5 elastic ip’s.

If we need more me have to request the aws official team.

* How to add the elastic ip:

1. Select the **EC2 instance**: **Network** **and Security**: **Elastic IP’s** : **Allocate the elastic IP**: Choose the **region**: **allocate**.
2. After allocation was done. Select the **elastic ip** and click on **actions** choose the **associate Elastic IP address**: choose **instance**: select the **server**: select the **private ip** of that server: **allocate**.

* How to remove the elastic ip:

1. Select the **instance**: **actions**: **network**: **Disassociate elastic ip address.**
2. After disassociate the elastic ip we have to release the Elastic ip.
3. Select **elastic ip**: **actions**: **release the elastic ip address**.