What is DevOps ?

DevOps is a set of practices, tools, and a cultural philosophy that automate and integrate the processes between software development and IT teams. It emphasizes team empowerment, cross-team communication and collaboration, and technology automation.

The DevOps lifecycle consists of eight phases representing the processes, capabilities, and tools needed for development . Phases include : Discover , Plan , Build , Test , Deploy , Operate , Observe , Continuous feedback

Phases of DevOps and tools used in DevOps ?

Discover Phase – gather idea and conduct research , **Mural and Miro**

Plan Phase - allow development and operations teams to break work down into smaller, manageable chunks for quicker deployments such as **Jira , Slack**

Build Phase - While Puppet and Chef primarily benefit operations, developers use open source tools like **Kubernetes and Docker** to provision individual development environments.

Continuous Delivery - Continuous integration is the practice of checking in code to a shared repository several times a day, and testing it each time. Tools include **Jenkins , Bitbucket , AWS**

Test - Testing tools span many needs and capabilities, including exploratory testing, test management, and orchestration.Testing tools span many needs and capabilities, including exploratory testing, test management, and orchestration.Tools are : **Xray , Mabl , Snyk**

Deploy - tools with a single dashboard integrated with your code repository and deployment tools.gives you full visibility on branches, builds, pull requests, and deployment warnings in one place.Tools include **BitBucket , AWSCodePipeline**

Operate - tools that integrate with your group chat client so alerts go straight to your team’s room, or a dedicated room for incidents.Tools include **Slack** , **Datadog**

Continuous feedback - applications that integrate your chat tool with your favorite survey platform for NPS-style feedback. Twitter and/or Facebook can also be integrated with chat for real-time feedback. Tools include **GetFeedback , Pendo**

Summary of tools :

1. Git and GitHub – Source code management (Version Control System)

2. Jenkins – Automation server, with plugins built for developing CI/ CD pipelines

3. Selenium – Automation testing

4. Docker – Software Containerization Platform

5. Kubernetes – Container Orchestration tool

6. Puppet – Configuration Management and Deployment

7. Chef – Configuration Management and Deployment

8. Ansible – Configuration Management and Deployment

9. Nagios – Continuous Monitoring

Roles and Responsibilities of a DevOps Engineer

Responsibilities of devops engineer include :

● Understanding customer requirements and project KPIs

● Implementing various development, testing, automation tools, and IT infrastructure

● Planning the team structure, activities, and involvement in project management activities.

● Managing stakeholders and external interfaces

● Setting up tools and required infrastructure

● Defining and setting development, test, release, update, and support processes for DevOps

operation

● Have the technical skill to review, verify, and validate the software code developed in the

project.

● Troubleshooting techniques and fixing the code bugs

Roles of DevOps engineer include :

1. DevOps Evangelist – The principal officer (leader) responsible for implementing DevOps

2. Release Manager – The one releasing new features & ensuring post-release product stability

3. Automation Expert – The guy responsible for achieving automation & orchestration of tools

4. Software Developer/ Tester – The one who develops the code and tests it

5. Quality Assurance – The one who ensures the quality of the product confirms to its requirement

6. Security Engineer – The one always monitoring the product's security & health

Git and Version Control System

By far, the most widely used modern version control system in the world today is Git. Git is a mature, actively maintained open source project originally developed in 2005 by Linus Torvalds, the famous creator of the Linux operating system kernel. A staggering number of software projects rely on Git for version control, including commercial projects as well as open source.

Developers who have worked with Git are well represented in the pool of available software development talent and it works well on a wide range of operating systems and IDEs (Integrated Development Environments).

Having a distributed architecture, Git is an example of a DVCS (hence Distributed Version Control System). Rather than have only one single place for the full version history of the software as is common in once-popular version control systems like CVS or Subversion (also known as SVN), in Git, every developer's working copy of the code is also a repository that can contain the full history of all changes.

What is version control system ?

Version control enables teams to collaborate and streamline development to resolve conflicts and create a centralized location for code.With version control, every change made to the code base is tracked. This allows software developers

to see the entire history of who changed what at any given time — and roll back from the current version to an earlier version if they need to. It also creates a single source of truth.

Version control (or source control or revision control) serves as a safety net to protect the source code from irreparable harm, giving the development team the freedom to experiment without fear of causing damage or creating code conflicts. Version control system can be of 4 types :

1)Distributed :

A distributed version control system (DVCS) allows users to access a repository from multiple locations. DVCSs are often used by developers who need to work on projects from multiple computers or who need to collaborate with other developers remotely.

2)Centralized :

A centralized version control system (CVCS) is a type of VCS where all users are working with the same central repository. This central repository can be located on a server or on a developer's local machine. Centralized version control systems are typically used in software development projects where a team of developers needs to share code and track changes.

3)Lock Based:

A lock-based version control system uses file locking to manage concurrent access to files and resources. File locking prevents two or more users from making conflicting changes to the same file or resource.

4)Optimistic:

In an optimistic version control system, every user has their own private workspace. When they want to share their changes with the rest of the team, they submit a request to the server. The server then looks at all the changes and determines which ones can be safely merged together.

git – version

git config –global user.Nj180280

What is Github and Github account?

GitHub is a for-profit company that offers a cloud-based Git repository hosting service. Essentially, it makes it a lot easier for individuals and teams to use Git for version control and collaboration.GitHub's interface is user-friendly enough so even novice coders can take advantage of Git. Without GitHub, using Git generally requires a bit more technical savvy and use of the command line.

Three types of github account –

- Personal accounts
- Organization accounts
- Enterprise accounts

Every person who uses GitHub signs into a personal account. An organization account enhances collaboration between multiple personal accounts, and an enterprise account allows central management of multiple organizations

Git Commands :

1) git config

Utility : To set your user name and email in the main configuration file.

How to : To check your name and email type in git config --global user.name and git config --global user.email.

2) git init

Utility : To initialise a git repository for a new or existing project.

How to : git init in the root of your project directory.

3) git clone

Utility : To copy a git repository from remote source, also sets the remote to original source so that you can pull again.

How to : git clone <:clone git url:>

4) git status

Utility : To check the status of files you've changed in your working directory, i.e, what all has changed since your last commit.

How to : git status in your working directory. lists out all the files that have been changed.

5) git add

Utility : adds changes to stage/index in your working directory.

How to : git add .

6) git commit

Utility : commits your changes and sets it to new commit object for your remote.

How to : git commit -m”sweet little commit message”

7) git push/git pull

Utility : Push or Pull your changes to remote. If you have added and committed your changes and you want to push them. Or if your remote has updated and you want those latest changes.

How to : git pull <:remote:> <:branch:> and git push <:remote:> <:branch:>

8) git branch

Utility : Lists out all the branches.

How to : git branch or git branch -a to list all the remote branches as well.

9) git checkout

Utility : Switch to different branches

How to : git checkout <:branch:> or **_git checkout -b <:branch:> if you want to create and switch to a new branch.

10) git stash

Utility : Save changes that you don’t want to commit immediately.

How to : git stash in your working directory. git stash apply if you want to bring your saved changes back.

11) git merge

Utility : Merge two branches you were working on.

How to : Switch to branch you want to merge everything in. git merge <:branch_you_want_to_merge:>

12) git reset

Utility : You know when you commit changes that are not complete, this sets your index to the latest commit that you want to work on with.

How to : git reset <:mode:> <:COMMIT:>

13) git remote

Utility : To check what remote/source you have or add a new remote.

How to : git remote to check and list. And git remote add <:remote_url:>

Commands –

git commit –global user.Nj180280

git commit –global email.niranjan180280@gmail.com

git init

git remote add origin https://github.com/Nj180280/temp.git

git status

git add “test1.txt”

git status

git commit -m “test1.txt added”

git pull origin main

git checkout -b branch1

git add “test1.txt”

ssh-keygen

git push origin

Jenkins

Jenkins is an open source automation server. With Jenkins, organizations can accelerate the software development process by automating it. Jenkins manages and controls software delivery processes throughout the entire lifecycle, including build, document, test, package, stage,deployment, static code analysis and much more. You can set up Jenkins to watch for any code changes in places like GitHub, Bitbucket or GitLab and automatically do a build a with tools like Maven and Gradle.

You can utilize containertechnology such as Docker and Kubernetes, initiate tests and then take actions like rolling backor rolling forward in production. Kohsuke was a developer at Sun and got tired of incurring the wrath of his team every time his code broke the build. He created Jenkins as a wayto perform continuous integration – that is, to test his code before he did an actual commit to the repository, to be sure all was well. Once his teammates saw what he was doing, they all wanted touse Jenkins. Kohsuke open sourced it, creating the Jenkins project, and soon Jenkins usage had spread around the world.

Originally developed for CI and now perform both CI/CD hence making software deployment fast , 1700 plugins

Jenkins Pipeline

A Jenkins Pipeline is a suite of plugins that supports implementing and integrating continuous delivery pipelines into Jenkins. It provides a way to define a sequence of build, test, and deployment actions as code, allowing you to automate and manage the entire software delivery process.

Jenkins Pipelines come in two flavors: Declarative and Scripted.

Declarative Pipeline: This is a more structured and opinionated way of defining your pipeline. It uses a simplified syntax and enforces some best practices. It's often easier to read and maintain.

Scripted Pipeline: This is more flexible and powerful, as it allows you to write your pipeline using Groovy script directly. It's suitable for more complex scenarios but can be harder to maintain.

```
pipeline {
    agent any
    stages {
        stage('Hello') {
            steps {
                echo 'Hello World'
            }
        }
```

```
    }
}
```

```
pipeline {
    agent any
    stages {
        stage('Build') {
        steps {
            echo 'Hi, GeekFlare. Starting to build the App.'
        }
        }
        stage('Test') {
        steps {
          input('Do you want to proceed?')
        }
        }
        stage('Deploy') {
        parallel {
                stage('Deploy start ') {
                steps {
                    echo "Start the deploy .."
                }
                }
                stage('Deploying now') {
                agent {
                        docker {
                            reuseNode true
                            image ‘nginx’
                            }
                        }

                    steps {
                      echo "Docker Created"
                    }
                }
                }
                }
          stage('Prod') {
             steps {
                      echo "App is Prod Ready"
                    }

        }
}
}
```

Docker

Docker is a popular platform for developing, shipping, and running applications in containers. Containers are lightweight, portable, and self-sufficient environments that contain all the necessary components to run an application, including the code, runtime, libraries, and dependencies. Docker provides a way to package applications and their dependencies into a standardized format, making it easier to deploy and manage software across different environments, such as development, testing, and production.

Docker Containers: Containers are instances of Docker images. An image is a lightweight, standalone, and executable package that includes everything needed to run an application, including the code, runtime, system tools, libraries, and settings.

Docker Engine: The Docker Engine is the core component of Docker that manages containers .It includes a server, REST API, and command-line interface (CLI) for interacting with Docker.

Dockerfile: A Docker file is a text file that contains a set of instructions for building a Docker image.

Docker Compose: Docker Compose is a tool for defining and running multi-container

Docker applications. It allows you to define the services, networks, and volumes in a single YAML file

Docker Hub: Docker Hub is a cloud-based registry service provided by Docker, Inc. It allows users to share and discover Docker images.

DOCKER ARCHITECTURE

Docker uses a client-server architecture that consists of several key components.

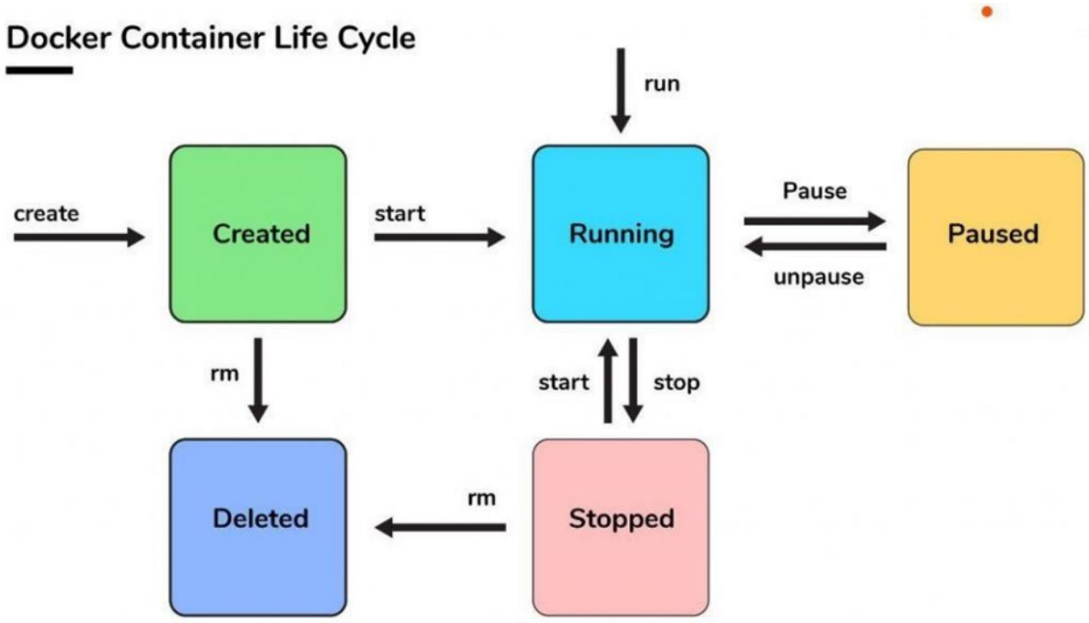
Docker Daemon (dockerd): The Docker daemon is a background service that runs on the host system. It is responsible for managing Docker containers, images, networks, and volumes.

Docker Client: The Docker client is a command-line tool or graphical user interface (GUI) that allows users to interact with the Docker daemon.

Docker Registry: Docker images are stored in registries, which are repositories for container images.

Docker Images: Docker images are read-only templates that contain an application's code, runtime, libraries, and dependencies.

Docker Containers: Containers are instances of Docker images that run in isolated environments on the host system.



Commands –

wsl –status

wsl –update

docker images

docker pull hello-world

docker run hello-world

docker ps (list all running containers)

docker ps -a(list all available containers)

steps to build sample web app in Docker :

1) **Choose a Base Image:** Start by selecting a base image that provides the foundational operating system and environment for your web application.
2) **Set working directory**
3) **Copy application files into docker image**
4) **Install dependencies**
5) **If web app listen to specific port , expose it**
6) **Define startup command like cmd**
7) **Build image**
8) **Run container**
9) **Access webapp**

Create sample image for web app using docker

1) Create a folder and create a docker file there and paste this inside Dockerfile
   FROM nginx:alpine
   COPY . /usr/share/nginx/docker/test
2) docker build -t html .
3) docker images
4) docker run -p 8080:80 -d html
5)

Nagios

Nagios is an open source IT system monitoring tool. It was designed to run on the Linux operating system and can monitor devices running Linux, Windows and Unix OSes.

Nagios software runs periodic checks on critical parameters of application, network and server resources. For example, Nagios can monitor memory use, disk use and microprocessor load, as well as the number of currently running processes and log files.

Nagios also can monitor services such as Simple Mail Transfer Protocol (SMTP), Post Office Protocol 3, Hypertext Transfer Protocol (HTTP) and other common network protocols.

Users can choose to work in the command-line interface or select a web-based graphical user interface in some versions of Nagios and from third parties. Based on the parameters and thresholds defined, Nagios can send out alerts if critical levels are reached.

Nagios runs both agent-based and agentless configurations. Independent agents are installed on any hardware or software system to collect data that is then reported back to the management server. Agentless monitoring uses existing protocols to emulate an agent.

Nagios can also run remote scripts and plugins using the Nagios Remote Plugin Executor (NRPE) agent. NRPE enables remote monitoring of system metrics such as system load, memory and disk use. There are around 50 plugins developed and maintained by Nagios, while there are over 3,000 from the community.

The service that was originally known as Nagios is now referred to as Nagios Core. Core is freely available as an open source monitoring software for IT systems, networks and infrastructure.

Puppet Tools

Puppet is an efficient system management tool for centralizing and automating the configuration management process. It can be used as a software deployment tool , It can also be utilized as open-source configuration management for server configuration, management, deployment, and orchestration. Puppet is specially designed to manage the configuration of Linux and Windows systems. It is written in Ruby and uses its unique Domain Specific Language (DSL) to describe system configuration.

Open Source Puppet

Puppet Enterprise – reporting , role based access , GUI etc

Things puppet can do

Configuration Management: Puppet allows you to define and manage the desired state of your infrastructure components (e.g., servers, applications, services) in a declarative manner.

Automation: Puppet automates the process of configuring and maintaining systems, reducing manual intervention and human error.

Infrastructure as Code (IaC): Puppet enables the practice of treating infrastructure as code, meaning you define and manage your infrastructure using code files.

Multi-Platform Support: Puppet supports various operating systems and can be used to manage both Unix/Linux and Windows environments.

Puppet has a primary-secondary node architecture.

The clients are distributed across the network and communicate with the primary-secondary environment where Puppet modules are present. The client agent sends a certificate with its ID to the server; the server then signs that certificate and sends it back to the client. This authentication allows for secure and verifiable communication between the client and the master.

Puppet Resources:

Puppet resources represent individual components or objects that you want to manage on your systems, such as files, packages, services, users, and groups. Resources are defined in Puppet code using a specific syntax

Puppet Classes:

Puppet classes are a way to organize and group related resources and configurations together. Classes are defined in Puppet code and provide a modular and reusable way to organize your infrastructure configurations.

Puppet Manifests:

Puppet manifests are files that contain Puppet code. They are used to define the configuration and desired state of your systems. Manifests typically have a .pp file extension and contain a collection of resource declarations, class includes, and other Puppet constructs.

Puppet Modules:

Puppet modules are a way to package and distribute Puppet code and configurations in a structured and reusable manner.

In Puppet, manifest files are used to define the desired configuration and state of your systems and infrastructure. Manifests are written in Puppet's domain-specific language (DSL), and they contain declarations of resources, classes, and other Puppet constructs.

Why Do We Need Puppet Manifest Files:

Declarative Configuration: Manifests allow you to declare the desired state of resources, making it clear what your infrastructure should look like.

Automation: Manifests automate the configuration of systems, reducing manual intervention and the potential for human error.

Consistency: Puppet ensures that configurations remain consistent across all nodes, even as they evolve or change.

Scalability: Puppet allows you to manage configurations for a large number of nodes simultaneously.

Version Control: Manifests can be version-controlled, enabling collaboration, change tracking, and rollback capabilities.

Software testing is done by the testers who test cases repeatedly to gain output according to the requirement in two ways: a) manually and b) automatically.

When the tester writes the testing script manually and tests the software until it functions properly known as manual testing & when this manual testing process becomes automatic, it can be defined as automatic testing.

Automation testing tools – SmartBear , TestComplete

Advantages –

1) It saves time and cost in testing and provides an increment in the efficiency of testing.

2) Automation testing improves the accuracy of testing

3) With automation, more cycles can be achieved

4) It also ensures consistency in testing

5) It's test scripts can be reusable

XPath stands for XML Path Language. It uses a non-XML syntax to provide a flexible way of addressing (pointing to) different parts of an XML document. It can also be used to test addressed nodes within a document to determine whether they match a pattern or not.

XPath is mainly used in XSLT, but can also be used as a much more powerful way of navigating through the DOM of any XML-like language document using XPath Expression, such as HTML and SVG, instead of relying on the Document.getElementById() or Document.querySelectorAll() methods, the Node.childNodes properties, and other DOM Core features.

Differences	Single Slash (/)	Double Slash (//)
Type of Path	Specifies an absolute path.	Specifies a relative path.
Node Selection	Selects only the immediate children of the current node.	Selects nodes at any level beneath the current node.
Scope	Limited to the immediate child nodes of the current node.	Includes all descendant nodes, regardless of their level.
Traversal Depth	It moves only one level down the hierarchy.	It traverses through all levels of the XML hierarchy.
Performance	Generally faster than double slash as it narrows down the search scope.	Relatively slower compared to single slash, especially for large XML documents.

What is Selenium? What are the Selenium suite components?

Selenium is a popular open-source software testing framework used for automating web applications. It is widely used for functional testing, regression testing, and performance testing.

Selenium supports multiple programming languages, including Java, C#, Python, and Ruby, making it accessible to a wide range of developers. Selenium provides several tools that allow developers to automate their tests, including: Selenium WebDriver: A tool for automating browser interactions and for controlling web browsers programmatically.

Components of Selenium Suite

selenium ide

- It is an open-source tool.
- Provide base, for extensions.
- It provides multi-browser support.
- No programming language experience is required while using Selenium IDE.

Selenium RC: RC stands for Remote Control. It allows the programmers to code in different programming languages like C#, Java, Perl, PHP, Python, Ruby, Scala, Groovy. The figure shows how the Remote Control Server works.

- It supports all web browsers.
- It can perform iteration and conditional operations.
- Execution is faster as compared to IDE.
- It has built-in test result generators.

Selenium Web Driver: Selenium Web Driver automates and controls initiated by the web browser. It does not rely on JavaScript for automation. It controls the browser directly by communicating with it. The figure shows how web driver works as an interface between Drivers and Bindings

- It directly communicates with the web browser.
- Execution is faster.
- It supports listeners.
- It supports IOS/Android application testing.

Selenium Grid: Basically, it is a server that allows the test to use a web browser instance running on remote machines. It provides the ability to run the test on a remote web browser, which helps to divide a load of testing across multiple machines and it will save enormous time.

Why Selenium ?

Open-Source Nature: Selenium is an open-source tool, which means it is free to use.

Cross-Browser Compatibility: Selenium supports various web browsers such as Chrome, Firefox, Internet Explorer, and Safari.

Support for Multiple Programming Languages: Selenium supports multiple programming languages including Java, Python, C#, Ruby, and others.

Parallel Test Execution: Selenium supports parallel test execution, which significantly reduces the time required to execute a large suite of test cases.

Support for Various Operating Systems: Selenium can be used on multiple operating systems, including Windows, macOS, and Linux, allowing for testing across different environments without limitations.